Generalized linear models

Mark Andrews

Contents

Introduction

The normal linear model that we described in Chapter 9 models an outcome variable as a normal distribution whose mean varies as a linear function of a set of predictors. As useful and important as this model is, it is clearly limited in its applicability to data that is both continuous and has a roughly (conditionally) normal distribution. That unquestionably excludes variables that are categorical, or other variables like count variables. We can, however, make relatively simple extensions to the normal linear model to produce a class of regression models that work in many respects just like the normal linear models but are applicable to data sets characterized by different types of outcomes variables such as categorical or count variables. These are known as the *generalized linear models*. As we will see through multiple examples, generalized linear models are defined by two features that distinguish them from normal linear models: they use non-normal distributions for the outcome variable, and use a *link function* that transforms the parameters of the outcome distribution so that the transformed parameter is a linear function of the predictor variables. In that sense, they make two relatively simple changes from the normal linear model, but by doing so, they can extend to a much wider range of problems.

In this chapter, we will cover most of the widely used examples of the generalized linear models, such the various logistic regression models, Poisson regression models, and so on. In each case, we will see how they are defined by their outcome variable distribution and an accompanying link model. We will also see how we may often mix and match different outcome distribution and link functions to produce different variants of the various models.

Binary logistic regression

Let use assume, as we did with the normal linear model, that we have n independent observations, that can be represented as the n pairs

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

Here, just as before, in each observation, the y_i is the observed value of a univariate *outcome* variable, and this is the variable which we are hoping to predict or explain. Also as before, each $\vec{x}_i$ is a row vector of values of a set of K predictor or explanatory variables that can predict or explain the value of the outcome variable. Now consider the situation where the outcome variable is binary variable. Any binary variable's values can be represented, without loss of generality, as $\{0, 1\}$. In other words, no matter what the actual values, e.g. $\{\texttt{no}, \texttt{yes}\}$, $\{\texttt{false}, \texttt{true}\}$, etc., we can always represent these by $\{0, 1\}$. If the outcome variable is a binary variable, we simply can not use the normal linear model here. To do so would make the highly implausible claim that each value of y_i was drawn from a normal distribution. Because the normal distribution is a unimodal, symmetric and continuous distribution, it can not be used as a probability distribution over the discrete values $\{0, 1\}$.

A suitable distribution for a binary valued random variable x is a Bernoulli distribution, an example of which we depict in Figure 1. A Bernoulli distribution is an extremely simple distribution. It has a single parameter, which we will denote by θ. This θ gives the probability that x takes the value of 1. In other words,

$$\mathrm{P}(x = 1) \doteq \theta,$$

and so the probability that x takes the value of 0, given that $\mathrm{P}(x = 0) = 1 - \mathrm{P}(x = 1)$, is $1 - \theta$.

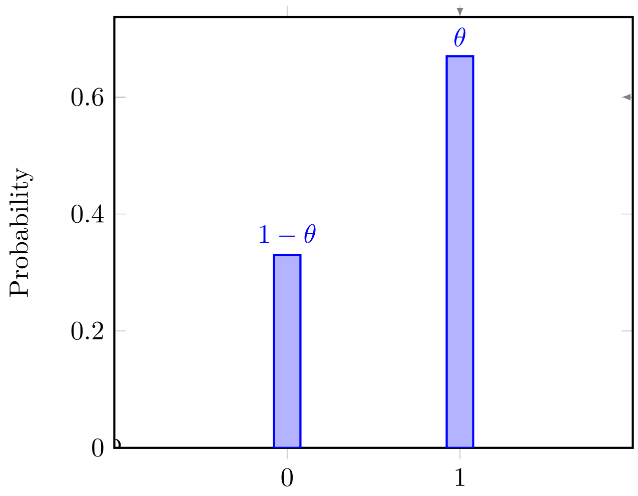


Figure 1: A Bernoulli distribution with parameter θ. The parameter θ gives the probability that the binary variable takes the value of 1. In other words, if x is a binary variable, $\mathrm{P}(x = 1) = \theta$, and so $\mathrm{P}(x = 0) = 1 - \mathrm{P}(x = 1) = 1 - \theta$.

We can therefore begin to extend the normal linear model by exchanging the normal distribution of the outcome variable for a Bernoulli distribution:

$$y_i \sim \mathrm{Bernoulli}(\theta_i), \quad \text{for } i \in 1 \ldots n.$$

In the case of the normal linear model, we had each $y_i \sim N(\mu_i, \sigma^2)$ and each μ_i being a linear function of $\vec{x}_i$, i.e. $\mu_i = \beta_0 + \sum_{k=1}^{K} \beta_k x_{ki}$. In the case of the Bernoulli distribution, however, we can not have each θ_i being a linear function of $\vec{x}_i$ because each θ_i is constrained to take values between 0 and 1, and in general, if we allow θ_i to be a linear function of $\vec{x}_i$, we can not guarantee that it will be constrained to the interval $(0, 1)$. In order to deal with this issue, we can transform θ_i to another variable ϕ_i that can take on any value on the real line $\mathbb{R}$ between $-\infty$ and ∞ and then treat ϕ_i as a linear function of $\vec{x}_i$. For this, we need an invertible function $f\colon (0,1) \mapsto \mathbb{R}$ that can map any value of θ_i to a unique value of ϕ, and vice versa. This function f is known as a *link function*, and it is a defining feature of a generalized linear model.

Our model extended model now is the following:

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

Compare this model to the original normal linear model, i.e.

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

There are two key differences here. First, the outcome variable's distribution is the normal distribution in the normal linear model, while it is the Bernoulli distribution for the case of binary outcome variable model. Second, in the normal linear model, the location parameter μ_i is a linear function of $\vec{x}_i$, while in the case of the binary outcome variable model, it is a transformation of the location parameter θ_i, rather than θ_i itself, that is the linear function of $\vec{x}_i$.

There are endless possibilities for the link function f, but the default choice, and in fact the defining choice for the binary logistic regression model is the *log odds*, otherwise known as the *logit*, function. The logit function is defined as

$$\phi = f(\theta) = \text{logit}(\theta) = \log_e \left(\frac{\theta}{1-\theta} \right).$$

In other words, this function takes a value $\theta \in (0,1)$ and divides it by $1 - \theta$, and then calculates the natural logarithm[1] of this function. The term $\frac{\theta}{1-\theta}$, when θ is assumed to be a probability, is known the *odds* of θ. Hence, the logit is simply the natural logarithm of the odds of θ. The logit function is invertible:

$$\theta = f^{-1}(\phi) = \text{ilogit}(\phi) = \frac{1}{1 + e^{-\phi}}.$$

This function is usually known as the inverse logit, hence ilogit, function. The logit and the inverse logit functions are shown in Figure 2a and Figure 2b, respectively.

The binary logistic regression model is therefore defined exactly as follows.

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

This can be written identically but in a more succinct manner as follows.

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

As an example of a binary logistic regression, let us look at a data set concerning extra marital affairs. This data set was conducted by the magazine *Psychology Today* and described in its July 1969 issue, and is described in more detail in Fair (1978).

```r
affairs_df <- read_csv('data/affairs.csv')
```

It has 601 observations for 9 variables. One of these 9 variables is `affairs`, which gives the number of times the respondent to the survey engaged in an extramarital act of sexual intercourse in the past year. The distribution of values of the `affairs` variable are as follows.

[1] Note that the natural logarithm is the logarithm to base $e \approx 2.7183$, hence we write it as $\log_e$. It is common to see this also written as ln. Because the natural logarithm is probably the most used logarithm base in statistics, we will usually denote it simply as log without explicitly stating the base.

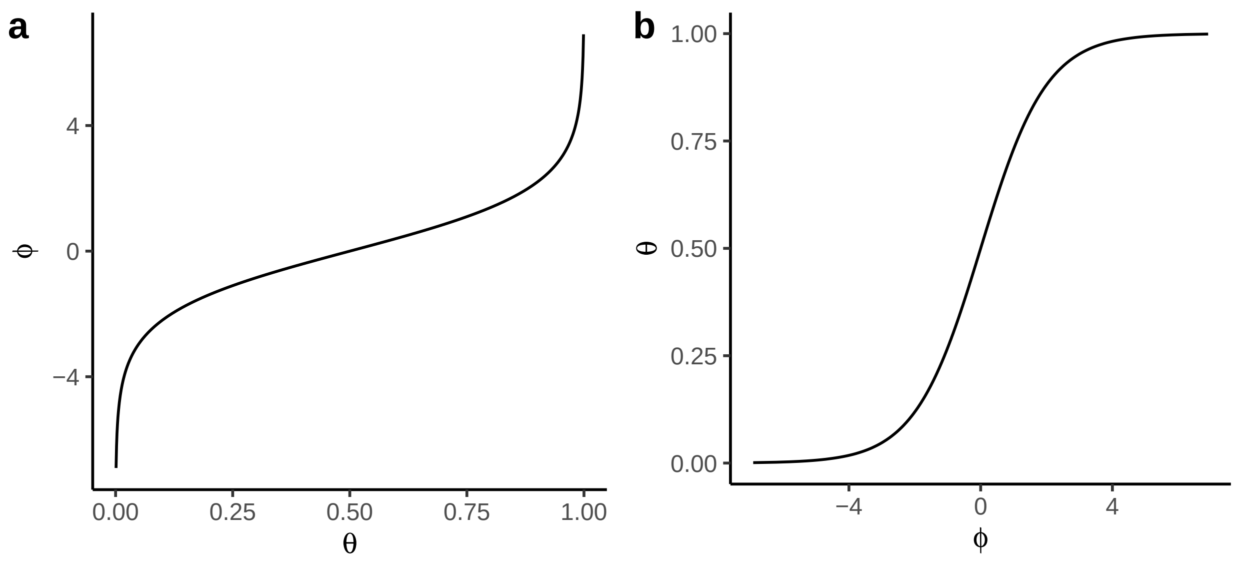


Figure 2: (a) The log odds, also known as the logit, function that maps the interval $(0, 1)$ to $\mathbb{R}$. (b) The inverse of the log odds, also known as the inverse logit, function that maps $\mathbb{R}$ to the interval $(0, 1)$.

```
affairs_df %>%
  pull(affairs) %>%
  table()
#> .
#>   0   1   2   3   7  12
#> 451  34  17  19  42  38
```

Here, the values of 0, 1, 2, and 3 indicate exactly 0, 1, 2, and 3 times, while 7 indicates 4-10 times, and 12 indicates monthly or weekly or daily. To simply matters, we will create a new variable `cheater` that takes the value of `TRUE` if the respondent engaged in any amount of extramarital sexual intercourse, and `FALSE` otherwise.

```
affairs_df %<>% mutate(cheater = affairs > 0)
```

This variable, which is obviously binary, will be our outcome variable. Other variables, which can serve as explanatory variables, include `gender` and `rating`. The `rating` variable has values of 1 to 5 that mean the following: 1 = very unhappy, 2 = somewhat unhappy, 3 = average, 4 = happier than average, 5 = very happy. In Figure 3 a-c, we show the proportion of people who cheat by a) gender b) marriage rating and c) marriage rating and gender.

We can understand binary logistic regression in a manner directly analogous to normal linear regression. Recall from Chapter 9 that we said that normal linear regression models the outcome variable as a normal distribution whose mean varies as we change the values of the predictor variables. In binary logistic regression, we model the outcome variable as a Bernoulli distribution whose parameter, which gives the probability of one of the two outcomes, varies as we change the values of the predictor variables. For example, consider Figure 3a. As we change `gender` from `female` to `male`, the proportion of those who have had affairs increases from 0.23 to 0.27. Similarly, as we see in Figure 3b, as `rating` increases, the proportion of people cheating declines. Likewise in Figure 3c, for females and males separately, as `rating` increases, the proportion of people cheating declines, but for the most part, for any given value of `rating`, the proportion of males who cheat is greater than the number of females who cheat. As we will see, we can model these changes in the probability of cheating as a function of predictors using a binary logistic regression analogously to how we could model changes in a normally distributed outcome variable as a function of predictors using normal

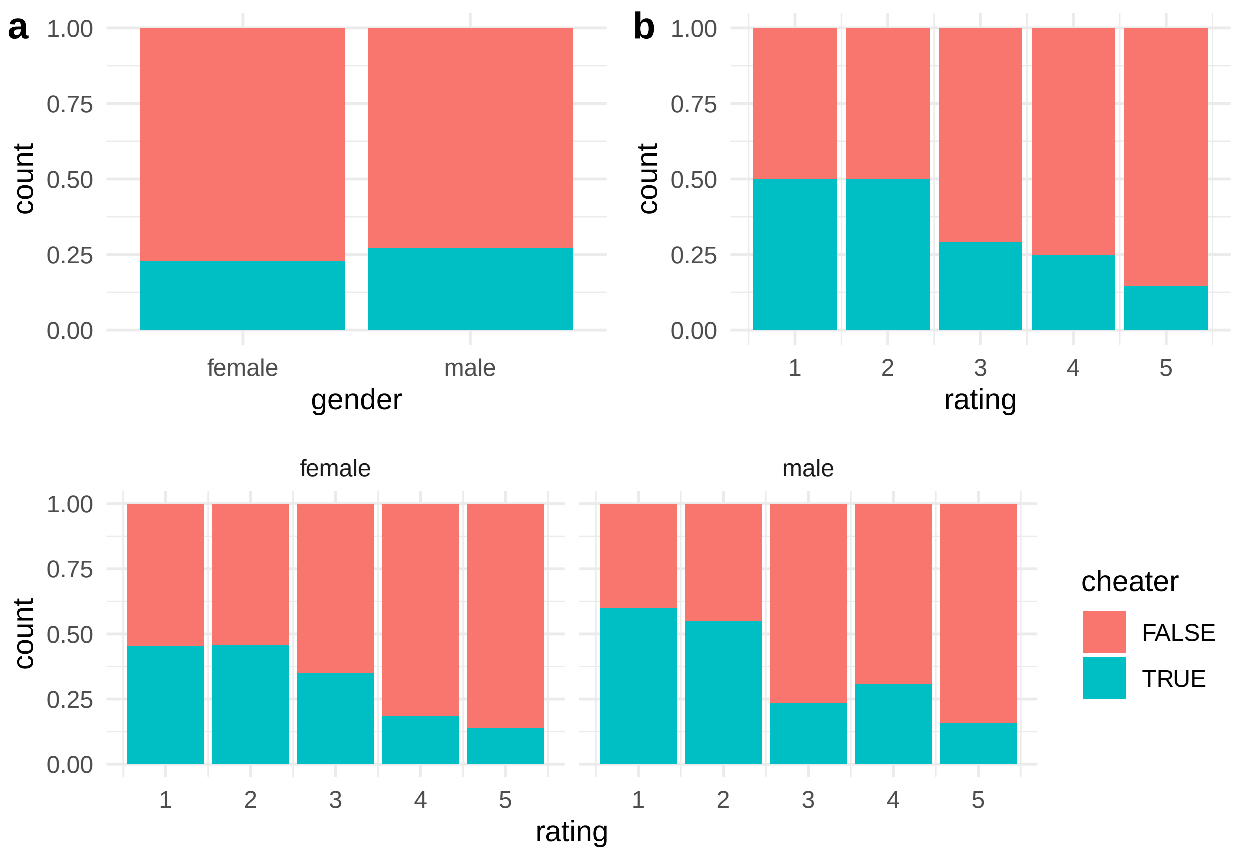


Figure 3: Each bar in each plot shows the proportion of people in the relevant group who have had an affair or not in the past year. (a) The proportions for female and males. (b) The proportions according to the rating of the happiness of the marriage. (c) The proportions according to the marriage rating for females and males.

linear regression.

One of the key differences between normal linear and binary logistic regression models, as we have mentioned, is that while in normal linear models, we model the location parameter of outcome variable as a linear function of the predictors, in binary logistic regression, we model the log odds of the location parameter of the outcome variable as a linear function of the predictors. In other words, in binary logistic regression, as predictor variable k changes by Δ_k, we assume the log odds of the probability of the outcome variable changes by $\beta_k \Delta_k$, where β_k is the coefficient for predictor k. In Figure 4a, we plot the proportion of cheaters as a function of the marriage rating level, and in Figure 4b, we plot the log odds of this proportion as a function of the marriage rating level.

Using the `affairs_df` data, we can model changes in the probability of cheating as a function of `gender` or `rating` or both `gender` and `rating` using a binary logistic regression as follows. When using `gender` as a predictor, our model would be as follows.

$$y_i \sim \mathrm{Bernoulli}(\theta_i), \quad \theta_i = \mathrm{ilogit}(\phi_i), \quad \phi_i = \beta_0 + \beta_1 x_{1i}, \quad \text{for } i \in 1 \ldots n,$$

or equivalently,

$$y_i \sim \mathrm{Bernoulli}(\theta_i), \quad \mathrm{logit}(\theta_i) = \beta_0 + \beta_1 x_{1i}, \quad \text{for } i \in 1 \ldots n,$$

where x_{1i} indicates if person i is a male or female by $x_{1i} = 1$ if $\texttt{gender}_i$ is male and $x_{1i} = 0$ if person $\texttt{gender}_i$ is female. Once we have inferred the value of β_0 and β_1, and we will describe how to do so below, $\mathrm{ilogit}(\beta_0 + \beta_1 \times 0) = \mathrm{ilogit}(\beta_0)$ will give us the estimate of the probability that a female will have an affair,

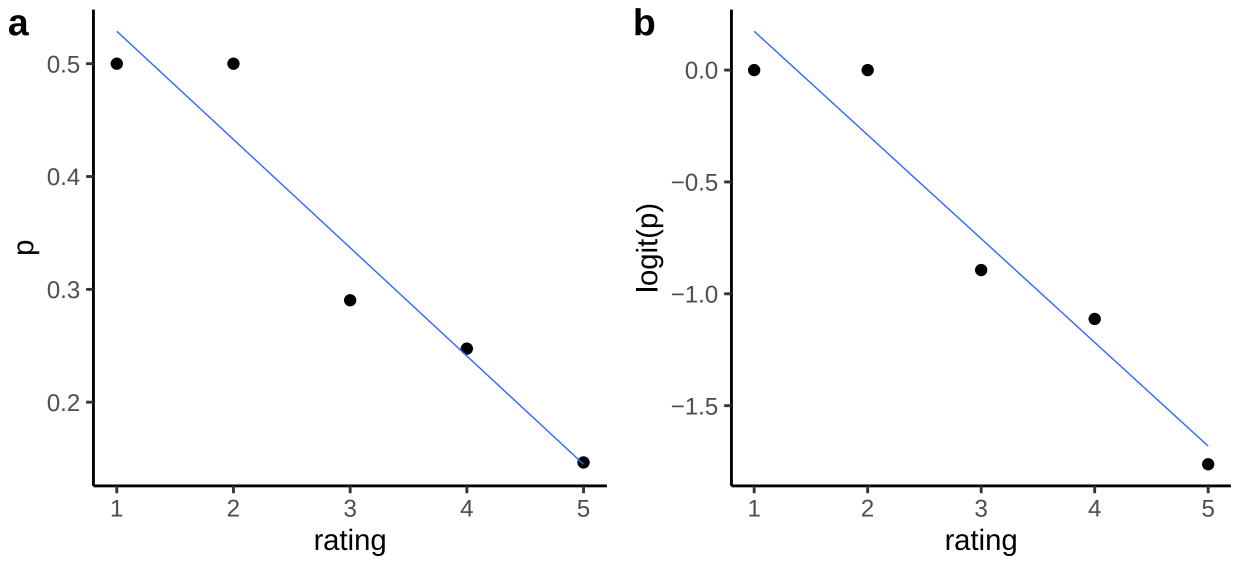


Figure 4: (a) The proportion of cheaters as a function of the marriage rating. (b) The log odds of the proportion of cheaters as a function of the marriage rating.

while $\text{ilogit}(\beta_0 + \beta_1 \times 1) = \text{ilogit}(\beta_0 + \beta_1)$ will give us the estimate of the probability that a male will have an affair. Equivalently, β_0 is the estimate of the log odds that a female will have an affair, while $\beta_0 + \beta_1$ is the estimate of the log odds that a male will have an affair. This means that β_1 is the difference in the log odds of having an affair between males and females. As we will discuss in more detail below, this also entails that e^{β_1} is the *odds-ratio* of having an affair between males and females.

If we wish to model how the probability of having an affair varies by the `rating` variable, our model could be the following

$$y_i \sim \text{Bernoulli}(\theta_i), \quad \theta_i = \text{ilogit}(\phi_i), \quad \phi_i = \beta_0 + \beta_2 x_{2i}, \quad \text{for } i \in 1 \ldots n,$$

which is equivalent to

$$y_i \sim \text{Bernoulli}(\theta_i), \quad \text{logit}(\theta_i) = \beta_0 + \beta_2 x_{2i}, \quad \text{for } i \in 1 \ldots n,$$

where $x_{2i} \in \{1, 2, 3, 4, 5\}$ is the rating of the happiness of the marriage by person i. Here, we are explicitly assuming that the log odds of having an affair varies linearly with a change in the value of `rating`. In other words, we assume that the log odds of having an affair changes by β_2 whenever `rating` changes by one unit, regardless if it changes from 1 to 2, 2 to 3, 3 to 4, or 4 to 5. That the log odds changes by this constant amount whenever `rating` changes by one unit is not strictly necessary, nor is it beyond dispute in this data set as we can see from Figure 4b. However, this assumption is a standard one in logistic regression generally, and to go beyond this assumption would require a nonlinear extension to the logistic regression, which is something we will not consider in this chapter.

Modelling how the probability of having an affair varies with `gender` and `rating`, assuming no interaction between these two variables, could be done with the following model

$$y_i \sim \text{Bernoulli}(\theta_i), \quad \theta_i = \text{ilogit}(\phi_i), \quad \phi_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i}, \quad \text{for } i \in 1 \ldots n,$$

which is equivalent to

$$y_i \sim \text{Bernoulli}(\theta_i), \quad \text{logit}(\theta_i) = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i}, \quad \text{for } i \in 1 \ldots n,$$

where x_{1i} and x_{2i} are as above. We interpret this model similarly to the two previous ones with the exception that now β_1 is the change in the log odds of having an affair as we go from females to males assuming that the value of `rating` is held constant. In other words, assuming that `rating` has any given value, as the log odds of having an affair changes by β_1 as we go from females to males. Likewise, holding `gender` constant, if `rating` increases by one unit, then the log odds of having an affair changes by β_2.

Maximum likelihood estimation

Like in normal linear regression, we can estimate the values of the unknown variables in the model, namely $\beta_0, \beta_1 \ldots \beta_K$ using maximum likelihood estimation. Unlike in the case of normal linear models, however, there is no closed form solution to the obtaining the maximum likelihood estimates. In other words, we can not simply solve for $\beta_0, \beta_1 \ldots \beta_K$ to find the values that maximize the likelihood function. Alternative, numerical methods to obtaining the maximum likelihood estimates are therefore used.

The likelihood function can be

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

where $\vec{y} = [y_1, y_2 \ldots y_n]^\intercal$, $\vec{\beta} = [\beta_0, \beta_1 \ldots \beta_K]^\intercal$, X is a matrix of n vertically stacked row vectors $\vec{x}_1, \vec{x}_2 \ldots \vec{x}_n$, and $\theta_i = \mathrm{ilogit}(X\vec{\beta})$. The logarithm of the likelihood is

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

Although this is clearly a function of $\vec{\beta}$, we can not, as we did in the case of normal linear models, simply calculate its gradient with respect to $\vec{\beta}$ and set it to zero and solve for $\vec{\beta}$. However, $L(\vec{\beta}|\vec{y}, X)$ is a convex function and has a global maximum , and hence we may use numerical optimization methods to find this global maximum. Relatively simple methods to maximize this function include gradient descent methods that choose an arbitrary starting value for $\vec{\beta}$, calculate the gradient of the function at this point, and then choose the next value of $\vec{\beta}$ by adding to it a constant times the gradient vector. More computationally efficient and effective methods include using Newton's method for root find applied to the derivative of the log of the likelihood function. When applied to binary logistic regression, this is known as *iteratively reweighted least squares* (see Murphy 2012 for details). Specifically, we start with an arbitrary starting value for $\vec{\beta}$, which we will call $\vec{\beta}_0$, and then for $t \in 0, 1, 2 \ldots$, we update our estimate of $\vec{\beta}$ using the following update rule until the estimate converges:

$$\vec{\beta}_{t+1} = (X^\intercal S_t X)^{-1} X^\intercal (S_t X \vec{\beta}_t + \vec{y} - \vec{\theta}_t).$$

Here, S_t is a $n \times n$ diagonal matrix whose value at the ith element of the diagonal is $\theta_i^t(1-\theta_i^t)$, where $\theta_i^t = X\vec{\beta}_t$, and $\vec{\theta}_t = [\theta_1^t, \theta_2^t \ldots \theta_n^t]^\intercal$.

As before, we will denote the maximum likelihood estimator of $\vec{\beta}$ by $\hat{\beta}$. It can be shown that the sampling distribution of $\hat{\beta}$ is distributed asymptotically as follows:

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

This result is very similar, though not identical, to the sampling distribution of $\hat{\beta}$ in the case of the normal linear model. Using this result, the sampling distribution for any particular coefficient is

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

which entails that

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

where $\sqrt{(X^\intercal S_t X)^{-1}_{kk}}$ is the standard error term.

Binary logistic regression using R

Using R, we can implement a binary logistic regression using the `glm` function. The `glm` function is used almost identically to how we used `lm`, but because it is for different types of generalized linear models and not just the binary logistic regression model, we must specify both the outcome variable probability distribution that we assume and also the link function.

When applied to the `affairs_df` problem, using `gender` and `rating` as the predictor variables, we implement the binary logistic regression in R as follows.

```
affairs_m <- glm(cheater ~ gender + rating,
                 family = binomial(link = 'logit'),
                 data = affairs_df)
```

As we can see, the way we use `glm` is almost identical to how we used `lm`, but we have to use a new argument, `family`, to specify the outcome distribution and link function. Given that the outcome variable's probability distribution is a Bernoulli distribution, it may seem unexpected to see that we state here that it is binomial distribution. However, the binomial distribution is in fact a generalization of the Bernoulli distribution; a binomial distribution when the number of observations is 1 is exactly the Bernoulli distribution. As such, it is technically correct to say that a binary variable has a binomial distribution. Note that we state the link function `link = 'logit'` inside `binomial()`. The logit link function is the default so we could simply write `family = binomial()`.

Just like with `lm`, we may see the maximum likelihood estimates of $\beta_0, \beta_1, \beta_2$ with the `coef()` function.

```
(estimates <- coef(affairs_m))
#> (Intercept)  gendermale      rating
#>   0.7093252   0.2547430  -0.5108324
```

From this, for example, we see that difference in the log odds of having an affair between males and females, assuming that `rating` is held constant at any value, is 0.255. Likewise, assuming `gender` is held constant, as we increase `rating` by one unit, the log odds of having an affair decreases by 0.511 (in other words, it increases by -0.511). The trouble with these statements about the coefficient is that they won't make much intuitive sense for those not used to thinking in terms of log odds. We will return to consider some alternative explanations of these coefficients below after we have considered predictions in logistic regression.

Let us now turn to hypothesis tests and confidence intervals for these coefficients. We may see the relevant information as follows.

```
summary(affairs_m)$coefficients
#>               Estimate Std. Error   z value     Pr(>|z|)
#> (Intercept)  0.7093252 0.33745388  2.101992 3.555402e-02
#> gendermale   0.2547430 0.19540750  1.303650 1.923529e-01
#> rating      -0.5108324 0.08495009 -6.013324 1.817574e-09
```

The standard error for each coefficient k is, as described above, $\hat{\mathrm{se}}_k = \sqrt{(X^\intercal S_t X)^{-1}_{kk}}$. We can confirm this easily similarly to how we did in the case of normal linear models.

```
library(modelr)

X <- model_matrix(affairs_df, cheater ~ gender + rating) %>%
  as.matrix()

p <- affairs_m$fitted.values
S <- diag(p * (1 - p))

(std_err <- solve(t(X) %*% S %*% X) %>% diag() %>% sqrt())
#> (Intercept)  gendermale      rating
#>  0.33745388  0.19540750  0.08495009
```

Note that `affairs_m$fitted.values` gives the values of $\vec{\theta}$ as defined above.

In the table above, the `z value` is the test statistics for the null hypothesis tests that the true values of each β_k are zero. In other words, it is $\hat{\beta}_k/\hat{\text{se}}_k$, as can be easily verified. The accompanying p-value, listed as `Pr(>|z|)`, is the probability of getting a result more extreme than the test statistic in a standard normal distribution.

```
z <- summary(affairs_m)$coefficients[,'z value']
2 * pnorm(abs(z), lower.tail = F)
#>  (Intercept)   gendermale       rating 
#> 3.555402e-02 1.923529e-01 1.817574e-09
```

The confidence intervals for the coefficients can be obtained as follows.

```
confint.default(affairs_m)
#>                   2.5 %     97.5 %
#> (Intercept)  0.04792776  1.3707227
#> gendermale  -0.12824866  0.6377347
#> rating      -0.67733153 -0.3443333
```

We can confirm that for each coefficient β_k, this is $\hat{\beta}_k \pm \hat{\text{se}}_k \cdot \zeta_{(0.975)}$, where $\zeta_{(0.975)}$ is the value below which lies 97.5% of the probability mass in a standard normal distribution. For example, for the case of `gender`, we have

```
estimates['gendermale'] + c(-1, 1) * std_err['gendermale'] * qnorm(0.975)
#> [1] -0.1282487  0.6377347
```

Predictions in binary logistic regression

Given $\hat{\beta}$, we can easily make predictions based on any given values of our predictors. In general, if $\vec{x}_\iota$ is a vector of values of the predictor variables that we wish to make predictions about, the predicted log odds corresponding to $\vec{x}_\iota$ is simply

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

and so the predicted probability of the outcome variable, which is the predicted values of the parameters of the Bernoulli distribution of the outcome variable, is

$$\theta_\iota = \frac{1}{1 + e^{-\phi_\iota}}.$$

For example, the predicted log odds of having an affair for a male with a `rating` value of 4 is as follows:

```
predicted_logodds <- (estimates['(Intercept)'] + estimates['gendermale'] * 1 + estimates['rating'] * 4)
  unname()
predicted_logodds
#> [1] -1.079261
```

If we then want the predicted probability, we use the inverse logit function. While this function does exist in R as the `plogis` function, it is nonetheless instructive to implement ourselves as it is a very simple function.

```
ilogit <- function(phi){
  1/(1 + exp(-phi))
}
```

Using this function, the predicted probability is as follows:

```
ilogit(predicted_logodds)
#> [1] 0.2536458
```

Doing predictions in logistic regression as we have just done is instructive but becomes tedious and error prone for all but very simple calculations. Instead, we should use the generic `predict` function as we did in the case of `lm`. For this, we must first set up a data frame with the same variables as are the predictors in the model and whose values are the values we want to make predictions about. For example, if we want to see the predictions for both females and males at all values of the `rating` variable, we can set up the data frame using `expand_grid`, which will give us all combinations of the values of the two variables.

```
affairs_df_new <- expand_grid(gender = c('female', 'male'),
                              rating = seq(5)
)
```

We can now make predictions as follows.

```
predict(affairs_m, newdata = affairs_df_new)
#>           1           2           3           4           5           6 
#>  0.19849280 -0.31233961 -0.82317202 -1.33400444 -1.84483685  0.45323579 
#>           7           8           9          10 
#> -0.05759662 -0.56842903 -1.07926144 -1.59009385
```

By default, this gives us the predicted log odds. We can get the predicted probabilities easily in one of two ways. First, we can pipe the predicted log odds to `ilogit`.

```
predict(affairs_m, newdata = affairs_df_new) %>% ilogit()
#>         1         2         3         4         5         6         7         8 
#> 0.5494609 0.4225438 0.3050907 0.2084978 0.1364803 0.6114083 0.4856048 0.3615994 
#>         9        10 
#> 0.2536458 0.1693707
```

Alternatively, we can use the `type = 'response'` argument with `predict`.

```
predict(affairs_m, newdata = affairs_df_new, type = 'response')
#>         1         2         3         4         5         6         7         8 
#> 0.5494609 0.4225438 0.3050907 0.2084978 0.1364803 0.6114083 0.4856048 0.3615994 
#>         9        10 
#> 0.2536458 0.1693707
```

As we have seen elsewhere, it is useful to use the `modelr::add_predictions` function to return these predictions as new variables in the data frame we are making predictions with.

```
affairs_df_new %>%
  add_predictions(affairs_m, type='response')
#> # A tibble: 10 x 3
#>    gender rating  pred
#>    <chr>   <int> <dbl>
#>  1 female      1 0.549
#>  2 female      2 0.423
#>  3 female      3 0.305
#>  4 female      4 0.208
#>  5 female      5 0.136
#>  6 male        1 0.611
#>  7 male        2 0.486
#>  8 male        3 0.362
#>  9 male        4 0.254
#> 10 male        5 0.169
```

Above, we established that the estimator $\hat{\beta}$ has the following asymptotic sampling distribution:

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

Given that the predicted log odds ϕ_ι is $\vec{x}_\iota \hat{\beta}$, the sampling distribution of ϕ_ι is as follows.

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

From this, the 95% confidence interval on the true value of ϕ_ι will be

$$\phi_\iota \pm \hat{\mathrm{se}}_\iota \cdot \zeta_{(0.975)}.$$

Unlike in the case of `lm`, there is no option for the `predict` function to return this confidence interval directly. However, it will return the standard errors, and from this, we can calculate the confidence intervals easily.

```
predictions <- predict(affairs_m, newdata = affairs_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  -0.31580817  0.71279377
#> 2  -0.69575622  0.07107699
#> 3  -1.11464246 -0.53170159
#> 4  -1.61390737 -1.05410150
#> 5  -2.20146018 -1.48821351
#> 6  -0.07157505  0.97804664
#> 7  -0.44875880  0.33356556
#> 8  -0.86174317 -0.27511488
#> 9  -1.35221279 -0.80631009
#> 10 -1.93420954 -1.24597816
```

The confidence intervals just given are on the log odds scale, but we can easily put them on the probability scale using the `ilogit` function.

```
cbind(
  predictions$fit - predictions$se.fit * qnorm(0.975),
  predictions$fit + predictions$se.fit * qnorm(0.975)
) %>% ilogit()
#>          [,1]      [,2]
#> 1  0.42169767 0.6710182
#> 2  0.33275380 0.5177618
#> 3  0.24700640 0.3701201
#> 4  0.16604683 0.2584383
#> 5  0.09961944 0.1841900
#> 6  0.48211387 0.7267205
#> 7  0.38965591 0.5826267
#> 8  0.29697528 0.4316518
#> 9  0.20550884 0.3086774
#> 10 0.12628538 0.2233971
```

Risk ratios and odds ratios

As mentioned above, the coefficients in a binary logistic regression give us differences in log odds. For example, we saw that the difference in the log odds of having an affair between males and females, assuming that `rating` is held constant at any value, is 0.255. We mentioned that these values are not easily interpreted in intuitive terms, and it is preferable to compare probabilities if possible.

Using the predicted probabilities we made above, we can `spread` the predictions for females and males to make them more easy to compare.

```
predictions <- affairs_df_new %>%
  add_predictions(affairs_m, type='response') %>%
  spread(gender, pred)
predictions
#> # A tibble: 5 x 3
#>   rating female  male
#>    <int>  <dbl> <dbl>
#> 1      1  0.549 0.611
#> 2      2  0.423 0.486
#> 3      3  0.305 0.362
#> 4      4  0.208 0.254
#> 5      5  0.136 0.169
```

With this, we can now calculate the difference and ratios of the probabilities of males and females.

```
predictions %>%
  mutate(prob_diff = male - female,
         prob_ratio = male/female)
#> # A tibble: 5 x 5
#>   rating female  male prob_diff prob_ratio
#>    <int>  <dbl> <dbl>     <dbl>      <dbl>
#> 1      1  0.549 0.611    0.0619       1.11
#> 2      2  0.423 0.486    0.0631       1.15
#> 3      3  0.305 0.362    0.0565       1.19
#> 4      4  0.208 0.254    0.0451       1.22
#> 5      5  0.136 0.169    0.0329       1.24
```

The `prob_ratio` values are obviously the ratios of the probabilities of having an affair by a males to the corresponding probabilities for females. These ratios are usually referred to as *risk ratios* or *relative risks*. Note, however, that these values are not constant across all values of `rating`. In other words, the relative risks of having an affairs by men and women varies according to value of `rating`.

Instead of ratios of probabilities, we can also calculate ratios of odds. We saw above that the odds is simply the ratio of a probability p to $1 - p$.

```
predictions %>%
  mutate(odds_male = male/(1-male),
         odds_female = female/(1-female),
         odds_ratio = odds_male/odds_female)
#> # A tibble: 5 x 6
#>   rating female  male odds_male odds_female odds_ratio
#>    <int>  <dbl> <dbl>     <dbl>       <dbl>      <dbl>
#> 1      1  0.549 0.611     1.57        1.22        1.29
#> 2      2  0.423 0.486     0.944       0.732       1.29
#> 3      3  0.305 0.362     0.566       0.439       1.29
#> 4      4  0.208 0.254     0.340       0.263       1.29
#> 5      5  0.136 0.169     0.204       0.158       1.29
```

As we can see, the odds ratios comparing males and females are constant for all values of `rating`. Thus, we can say that for any value of `rating`, the odds of having an affair by a man is exactly 1.29 greater than the odds of having an affair by a female.

Let us look more closely at how we calculated these odds ratios. Let us assume that the value of `rating` is r. Then, the log odds of having an affair by a female and a male are, respectively,

$$\log\left(\frac{\theta_{\text{female}}}{1-\theta_{\text{female}}}\right) = \beta_0 + \beta_2 \cdot r, \quad \log\left(\frac{\theta_{\text{male}}}{1-\theta_{\text{male}}}\right) = \beta_0 + \beta_1 + \beta_2 \cdot r.$$

This means that the odds of having an affair by a female or a male are, respectively,

$$\frac{\theta_{\text{female}}}{1-\theta_{\text{female}}} = e^{\beta_0+\beta_2\cdot r}, \quad \frac{\theta_{\text{male}}}{1-\theta_{\text{male}}} = e^{\beta_0+\beta_1+\beta_2\cdot r}.$$

The odds ratio comparing males to females is therefore

$$\frac{\theta_{\text{male}}}{1-\theta_{\text{male}}} \Bigg/ \frac{\theta_{\text{female}}}{1-\theta_{\text{female}}} = \frac{e^{\beta_0+\beta_1+\beta_2\cdot r}}{e^{\beta_0+\beta_2\cdot r}} = e^{(\beta_0+\beta_1+\beta_2\cdot r)-(\beta_0+\beta_2\cdot r)} = e^{\beta_1}.$$

More generally, by the same reasoning, we can see that for any predictor k, e^{β_k} gives the odds ratio corresponding to a unit change in x_k. In other words, e^{β_k} is the factor by which the odds increases whenever x_k increases by one unit, assuming any other predictors are held constant.

Note that as we saw above, we can obtain the 95% confidence intervals for the coefficients as follows.

```
confint.default(affairs_m, parm = c('gendermale', 'rating'))
#>                  2.5 %     97.5 %
#> gendermale -0.1282487  0.6377347
#> rating     -0.6773315 -0.3443333
```

To get the confidence intervals on the odds ratios corresponding to these predictors, we simply raise the confidence intervals to the power of e.

```
confint.default(affairs_m, parm = c('gendermale', 'rating')) %>%
  exp()
#>                2.5 %    97.5 %
#> gendermale 0.8796346 1.8921896
#> rating     0.5079707 0.7086927
```

Model comparison

As we saw above, we obtain the p-values for the null hypothesis tests that the true values of the coefficients are zero from the z statistic that is the estimate of the coefficient divided by its standard error. A more general way of doing null hypothesis tests in logistic regression, and also in related models as we will see below, is to perform a log likelihood ratio test. This allows us to compare one model with K predictors to another model with $K' < K$ predictors, where the K' predictors are a subset of the K predictors. This is a type of *nested model comparison* because the model with the K' predictors is a subset of the model with the K predictors. For example, we could compare the model using the `gender` and the `rating` predictors to a model using either `gender` or `rating` alone, or to a model using no predictors. In each of these two comparisons, we are comparing a model with two predictors with a model with a subset of these two predictors.

Generally speaking, we can describe the problem of nested model comparison using binary logistic regressions as follows. We assume, as before, that our outcome variable is $y_1, y_2 \ldots y_i \ldots y_n$, where each $y_i \in \{0, 1\}$, and that we have a set of predictors $\vec{x}_1, \vec{x}_2 \ldots \vec{x}_i \ldots \vec{x}_n$, where each $\vec{x}_i$ is

$$\vec{x}_i = x_{1i}, x_{2i} \ldots x_{ki} \ldots x_{K'i} \ldots x_{Ki}.$$

Obviously, $x_{1i}, x_{2i} \ldots x_{ki} \ldots x_{K'i} \subset x_{1i}, x_{2i} \ldots x_{ki} \ldots x_{K'i} \ldots x_{Ki}$.

From this, we can set up two models, one nested in the other. The first model, which we will call $\mathcal{M}_1$, uses all K predictors.

$$\mathcal{M}_1\colon y_i \sim \text{Bernoulli}(\theta_i), \quad \text{logit}(\theta_i) = \beta_0 + \sum_{k=1}^{K} \beta_k x_{ki}.$$

We will compare this to model $\mathcal{M}_0$ that uses K' predictors.

$$\mathcal{M}_0\colon y_i \sim \text{Bernoulli}(\theta_i), \quad \text{logit}(\theta_i) = \beta_0 + \sum_{k=1}^{K'} \beta_k x_{ki}.$$

The null hypothesis comparing $\mathcal{M}_1$ and $\mathcal{M}_0$ is that

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

In other words, it is the hypothesis that all the coefficients corresponding to the predictors that are in $\mathcal{M}_1$ but not in $\mathcal{M}_0$ are simultaneously zero. We can test this null hypothesis using a *likelihood ratio test*.

We begin by inferring the maximum likelihood estimators of the coefficients in both $\mathcal{M}_0$ and $\mathcal{M}_1$. We will denote the estimators for $\mathcal{M}_0$ and $\mathcal{M}_1$, by $\hat{\beta}_{\mathcal{M}_0}$ and $\hat{\beta}_{\mathcal{M}_1}$. Having done so we can obtain the value of the likelihood function in $\mathcal{M}_0$ and $\mathcal{M}_1$ evaluated at $\hat{\beta}_{\mathcal{M}_0}$ and $\hat{\beta}_{\mathcal{M}_1}$. We will denote these by $\mathcal{L}_0$ and $\mathcal{L}_1$, respectively. The likelihood ratio comparing $\mathcal{M}_0$ to $\mathcal{M}_1$ is simply

$$\text{likelihood ratio} = \frac{\mathcal{L}_0}{\mathcal{L}_1}.$$

The logarithm of this likelihood is

$$\log \text{likelihood ratio} = \log\left(\frac{\mathcal{L}_0}{\mathcal{L}_1}\right) = \log(\mathcal{L}_0) - \log(\mathcal{L}_1).$$

According to Wilks' theorem, when the null hypothesis is true, $-2 \times \log \text{likelihood ratio}$ is asymptotically distributed as a χ^2 distribution with degrees of freedom $K - K'$. Therefore, we calculate $-2 \times \log \text{likelihood ratio}$ and the calculate the p-value, which is simply the probability of a getting a result greater than $-2 \times \log \text{likelihood ratio}$ in a χ^2 distribution with degrees of freedom $K - K'$. Because

$$\log \text{likelihood ratio} = \log(\mathcal{L}_0) - \log(\mathcal{L}_1),$$

we have

$$-2 \times \log \text{likelihood ratio} = (-2 \cdot \log(\mathcal{L}_0)) - (-2 \cdot \log(\mathcal{L}_1)).$$

We refer to -2 times the log of the likelihood of a model as its *deviance*, and we'll denote the deviances of models $\mathcal{M}_0$ and $\mathcal{M}_1$ by $\mathcal{D}_0$ and $\mathcal{D}_1$, respectively:

$$\mathcal{D}_0 = -2 \cdot \log(\mathcal{L}_0), \quad \mathcal{D}_1 = -2 \cdot \log(\mathcal{L}_1).$$

Therefore, our likelihood ratio based null hypothesis test is based on the statistic

$$\mathcal{D}_0 - \mathcal{D}_1,$$

that we compare to a χ^2 distribution with $K - K'$ degrees of freedom.

We can perform this null hypothesis test in R easily in different ways. As an example, we will compare the model with the two predictors `gender` and `rating` to a model with neither. We already have the model with both predictors, and have named it `affairs_m`. We name the model with neither predictor `affairs_m0`.

```r
affairs_m0 <- glm(cheater ~ 1,
                  family = binomial(link = 'logit'),
                  data = affairs_df)
```

The formula `cheater ~ 1` indicates that we have an intercept only in this model and so the model is

$$y_i \sim \text{Bernoulli}(\theta_i), \quad \text{logit}(\theta_i) = \beta_0,$$

which entails that we assume that for each observation, there is a fixed probability, namely $\text{ilogit}(\beta_0)$, that $y_i = 1$.

We can obtain the log of the likelihoods of `affairs_m` and `affairs_m0` with the `logLik` function.

```
logLik(affairs_m)
#> 'log Lik.' -318.2889 (df=3)
logLik(affairs_m0)
#> 'log Lik.' -337.6885 (df=1)
```

The corresponding deviances can be obtained with the `deviance` function.

```
deviance(affairs_m)
#> [1] 636.5778
deviance(affairs_m0)
#> [1] 675.377
```

These are easily verified as -2 times the log of the likelihoods.

```
logLik(affairs_m) * -2
#> 'log Lik.' 636.5778 (df=3)
logLik(affairs_m0) * -2
#> 'log Lik.' 675.377 (df=1)
```

The difference of the two deviances is as follows.

```
deviance(affairs_m0) - deviance(affairs_m)
#> [1] 38.79919
```

If the null hypothesis is true, this difference of the deviances will be distributed as a χ^2 distribution with 2 degrees of freedom. The p-value for the null hypothesis is therefore

```
K <- affairs_m %>% coef() %>% length()
K_prime <- affairs_m0 %>% coef() %>% length()
pchisq(deviance(affairs_m0) - deviance(affairs_m),
       df = K - K_prime,
       lower.tail = F)
#> [1] 3.757193e-09
```

While it is instructive to go through the calculations in a step by step manner as we have just done, in practice it is much easier and less error prone to use the generic `anova` function for doing likelihood ratio tests. We perform the above analyses using `anova` as follows.

```
anova(affairs_m0, affairs_m, test='Chisq')
#> Analysis of Deviance Table
#>
#> Model 1: cheater ~ 1
#> Model 2: cheater ~ gender + rating
#>   Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
#> 1       600     675.38
#> 2       598     636.58  2   38.799 3.757e-09 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

As we can see, from this output, we have the deviances, the differences of the deviance, the degrees of freedom for the χ^2 distribution, and the p-value.

Bayesian approaches to logistic regression

The Bayesian approach to binary logistic regression begins with an identical probabilistic model to that of the classical approach In other words, we assume our data is

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

where each $y_i \in \{0, 1\}$ and each $\vec{x}_i$ is a vector of the values of K predictor or explanatory variables, and that

$$y_i \sim \text{Bernoulli}(\theta_i), \quad \text{logit}(\theta_i) = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i}, \quad \text{for } i \in 1 \ldots n.$$

At this point, the classical approach and the Bayesian approach diverge. The classical approach obtains the maximum likelihood estimators $\hat{\beta}$ and uses these estimators and their sampling distribution for hypothesis testing, confidence intervals, and predictions, as we have seen above. On the other hand, the Bayesian approach begins with the essential step of inferring the posterior distribution:

$$\overbrace{\mathrm{P}(\vec{\beta}|X, \vec{y})}^{\text{posterior}} = \frac{\overbrace{\mathrm{P}(\vec{y}|X, \vec{\beta})}^{\text{likelihood}}\overbrace{\mathrm{P}(\vec{\beta})}^{\text{prior}}}{\underbrace{\int \mathrm{P}(\vec{y}|X, \vec{\beta})\mathrm{P}(\vec{\beta})d\vec{\beta},}_{\text{marginal likelihood}}} \propto \overbrace{\mathrm{P}(\vec{y}|X, \vec{\beta})}^{\text{likelihood}}\overbrace{\mathrm{P}(\vec{\beta})}^{\text{prior}}$$

where $\vec{y} = y_1, y_2 \ldots y_n$, X is the matrix whose rows are $\vec{x}_1, \vec{x}_2, \ldots \vec{x}_n$, $\vec{\beta} = \beta_0, \beta_1 \ldots \beta_K$. The likelihood function, which we have seen above, is a function over $\vec{\beta}$. Its value gives us the probability of the observing our data given any value of $\vec{\beta}$. The prior, on the other hand, is also a function over $\vec{\beta}$, specifically a probability density function. It gives the probability distribution over the possible values that $\vec{\beta}$ could take in principle. These two functions are multiplied by one another to result in a new function over $\vec{\beta}$, which is then divided by its integral so that the resulting posterior distribution integrates to one, and hence is a probability density function. We interpret the posterior distribution as follows. Assuming that the data is generated by the stated logistic regression model, and also that the possible values that the true $\vec{\beta}$ could take in principle are given by $\mathrm{P}(\vec{\beta})$, then the posterior distribution gives the probability that the true value of $\vec{\beta}$ is any given value.

Unlike the case of Bayesian linear regression, there are no choices of prior that will lead to an analytic or closed form solution to the posterior distribution. As such, we must use alternative numerical methods. One traditionally commonly used approach, described in Bishop (2006), Murphy (2012) and elsewhere is to use a *Laplace approximation* to the posterior distribution, which approximates the posterior distribution is a multivariate normal distribution. However, given the current state of general purpose software for MCMC sampling in Bayesian models, as we describe in Chapter 8, it is now practically much easier to use MCMC methods, particularly the Hamiltonian Monte Carlo methods available with the Stan probabilistic programming language. As we've seen, a very easy to use R based interface to Stan is available through the `brms` package.

In the following code, we define and fit a Bayesian logistic regression model predicting `cheater` from both `gender` and `rating`, just as we did above.

```
affairs_m_bayes <- brm(cheater ~ gender + rating,
                       family = bernoulli(),
                       data = affairs_df)
```

This syntax is almost identical to the `glm` model. However, the `family` is specified as `bernoulli` rather than `binomial`. The link function will default to `logit`.

By using the default settings, we use 4 chains, each with 2000 iterations, and where the initial `S$warmup` iterations are discarded, leaving to 4000 total samples. The priors used by default are seen in the following table.

```
prior_summary(affairs_m_bayes)
#>                 prior     class       coef group resp dpar nlpar bound
#> 1                             b
#> 2                             b gendermale
#> 3                             b     rating
#> 4 student_t(3, 0, 10) Intercept
```

The blank in the `prior` column for the coefficients for `gender` and `rating` tell us that a uniform prior is being used, while a non-standard t-distribution with 3 degrees and a scale of 10 is on the `Intercept` coefficient.

We can view the summary of the inference of the coefficients as follows.

```
summary(affairs_m_bayes)$fixed
#>              Estimate  Est.Error   l-95% CI   u-95% CI     Rhat Bulk_ESS
#> Intercept   0.7207196 0.33687018  0.0523234  1.3808567 1.001145     4176
#> gendermale  0.2510777 0.19218312 -0.1149891  0.6208058 1.000614     3786
#> rating     -0.5148690 0.08407121 -0.6798001 -0.3485882 1.001154     3434
#>            Tail_ESS
#> Intercept      2889
#> gendermale     3184
#> rating         3372
```

First, the `Rhat` values are all almost exactly equal to 1.0, which indicates that the chains have converged. The `Bulk_ESS`[2] is an estimate of the number of independent samples that are informationally equivalent to the samples from the sampler, which are necessarily non-independent. We see that for each coefficient these are close to the theoretical maximum of 4000. This indicates that the sampler is very efficient.

Notice how the posterior mean and its standard deviation, given by `Estimate` and `Est.Error`, respectively, are almost identical to the maximum likelihood estimator and the standard error of the sampling distribution of the maximum likelihood estimator. Likewise, the 95% credible interval, given by `l-95% CI` and `u-95% CI` also closely parallel the classical 95% confidence intervals. We saw this close parallel between the classical and Bayesian models in the case of linear regression as well. It is to be expected in any situation where we have a relatively large amount of data, thus leading to a concentrated likelihood function, and a diffuse prior distribution.

In the following code, we calculate the 95% posterior interval on $\phi_\iota = \vec{x}_\iota \vec{\beta}$, where $\vec{x}_\iota$ is a vector of values of the predictor variables. We do this for each observation in the `affair_df_new` data frame that used above.

```
posterior_linpred(affairs_m_bayes, newdata = affairs_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(affairs_df_new, .)
#> # A tibble: 10 x 5
#>    gender rating `l-95% CI` prediction `u-95% CI`
#>    <chr>   <int>      <dbl>      <dbl>      <dbl>
#>  1 female      1    -0.315       0.209      0.723
#>  2 female      2    -0.698      -0.309      0.0735
#>  3 female      3    -1.12       -0.821     -0.529
#>  4 female      4    -1.62       -1.34      -1.06
#>  5 female      5    -2.22       -1.85      -1.51
#>  6 male        1    -0.0657      0.457      0.969
#>  7 male        2    -0.444      -0.0546     0.338
#>  8 male        3    -0.863      -0.567     -0.287
#>  9 male        4    -1.36       -1.09      -0.817
#> 10 male        5    -1.95       -1.60      -1.26
```

Again, we see a close parallel between these results and those of the 95% confidence interval on predictions obtained from the classical approach.

[2]The `Bulk_ESS` and `Tail_ESS` are two separate measures of effective samples size, with one (`Bulk`) using samples from the center of the distribution of the samples and the other (`Tail`) using samples from the tails. We will primarily focus on `Bulk_ESS` here, but if `Tail_ESS` is low when `Bulk_ESS` is not low, which may happen in heavy tailed distribution, this may indicate convergence problems with the sampler.

Poisson regression

Let us now consider another data analysis problem where we again have n independent observations represented as the n pairs

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

and where, as before, each $\vec{x}_i$ is a row vector of values of a set of K explanatory variables, but where now $y_i \in 0, 1, 2, \ldots$. In other words, in this problem, each y_i is neither the observed value of a real valued variable nor of a binary variable. Rather, each 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. 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 a non-negative integer: it can take values of 0, but not negative values, and it must be a whole number.

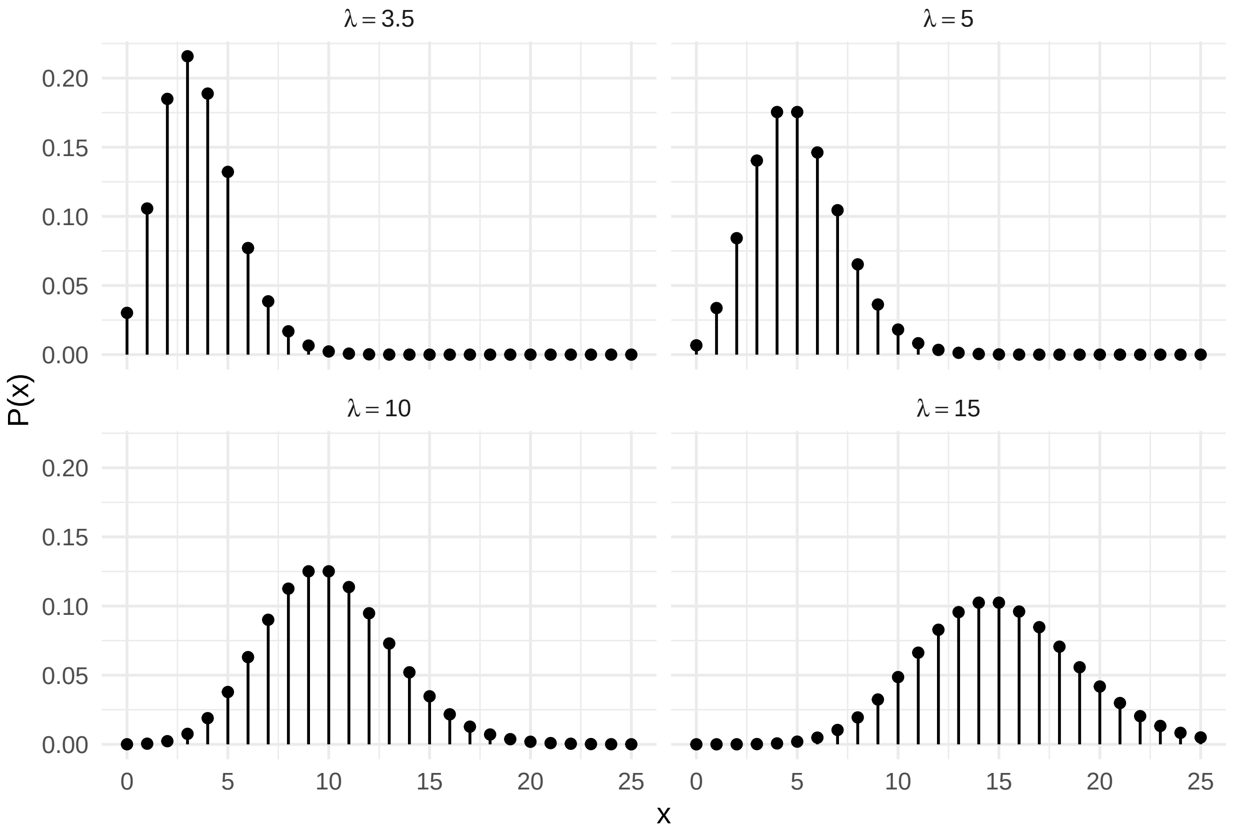


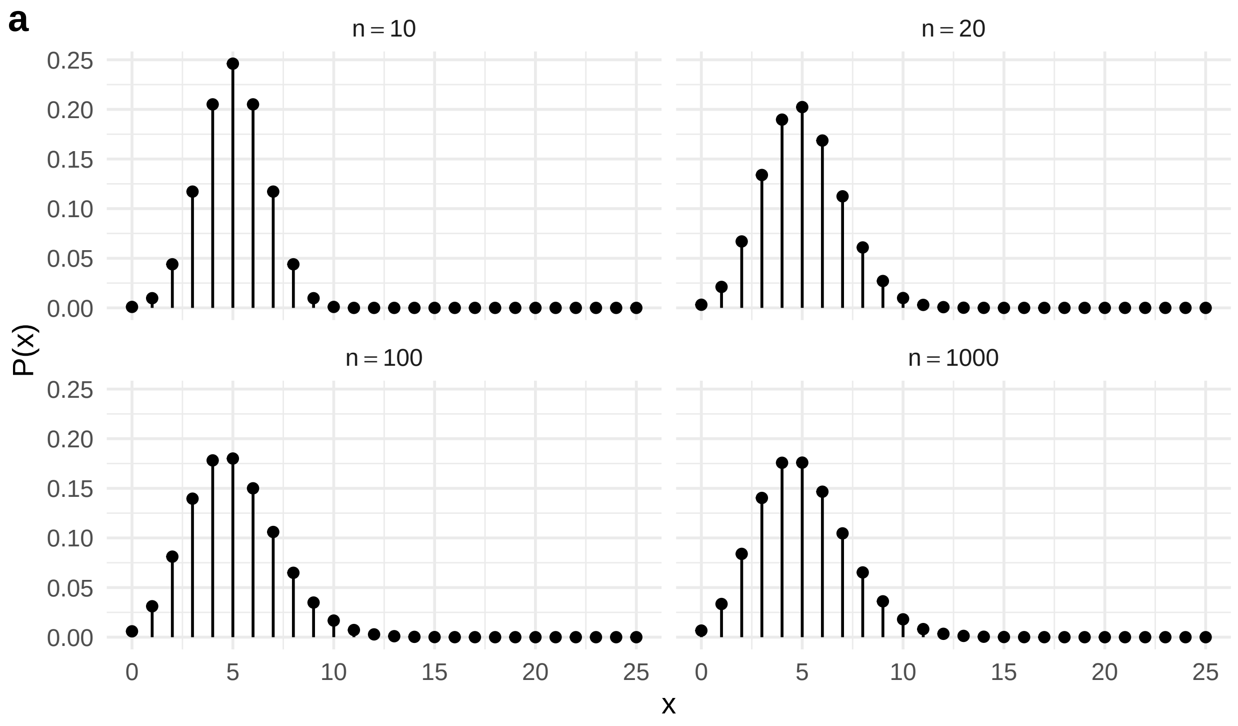
Figure 5: Poisson distributions with different values of the parameter λ.

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 $k \in 0, 1, \ldots$, and the probability that $x = k$ is

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

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 5, 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 6.



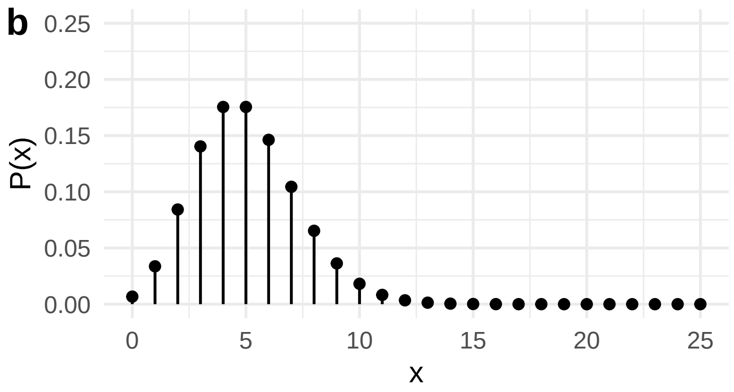


Figure 6: a) Binomial distributions with increasing values of n, which denote the number of trials, but where $\theta \cdot n$ is held constant at $\theta \cdot n = 5$. b) A Poisson distribution with $\lambda = 5$.

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:

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

which is identical to

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

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')
```

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 7, 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.

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 $\vec{y}$ and X are defined as in the case of linear and logistic regression. The logarithm of the likelihood is then defined as

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

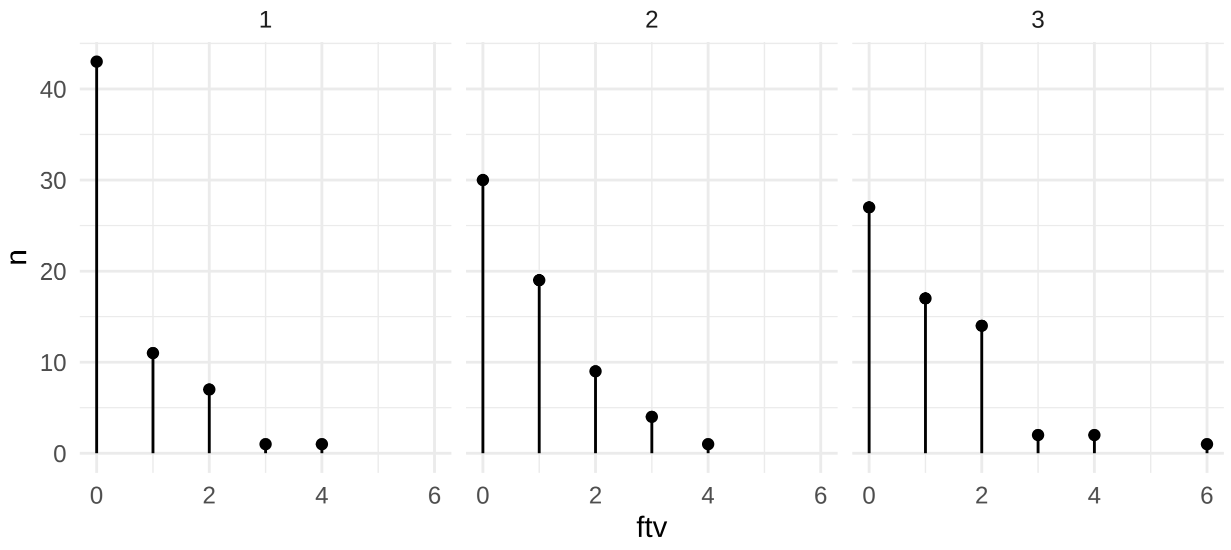


Figure 7: The number of visits to a doctor in the first trimester of pregnancy for each age tercile.

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

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

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

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

where $\sqrt{(X^{\intercal}WX)^{-1}_{kk}}$ 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.

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.

```
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 $\hat{\beta}_k \pm \hat{\mathrm{se}}_k \cdot \zeta_{(0.975)}$.

```
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

```r
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.

```r
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`.

```r
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.

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

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

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

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

```r
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.

```r
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.

```r
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.

```r
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 $\mathcal{L}_0$.

```r
-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

```r
(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.

```r
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.

```r
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. 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

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

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

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

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)
```

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, 10) 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 `S$warmup` 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
#> Intercept -1.41100632 0.3579772 -2.12522532 -0.73184557 1.001225     2667
#> age        0.04892008 0.0140871  0.02157362  0.07632476 1.000911     2952
#>           Tail_ESS
#> Intercept     2520
#> age           2663
```

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     -0.652    -0.431     -0.232
#> 2    25     -0.352    -0.185     -0.0331
#> 3    30     -0.162     0.0581     0.269
```

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 8.

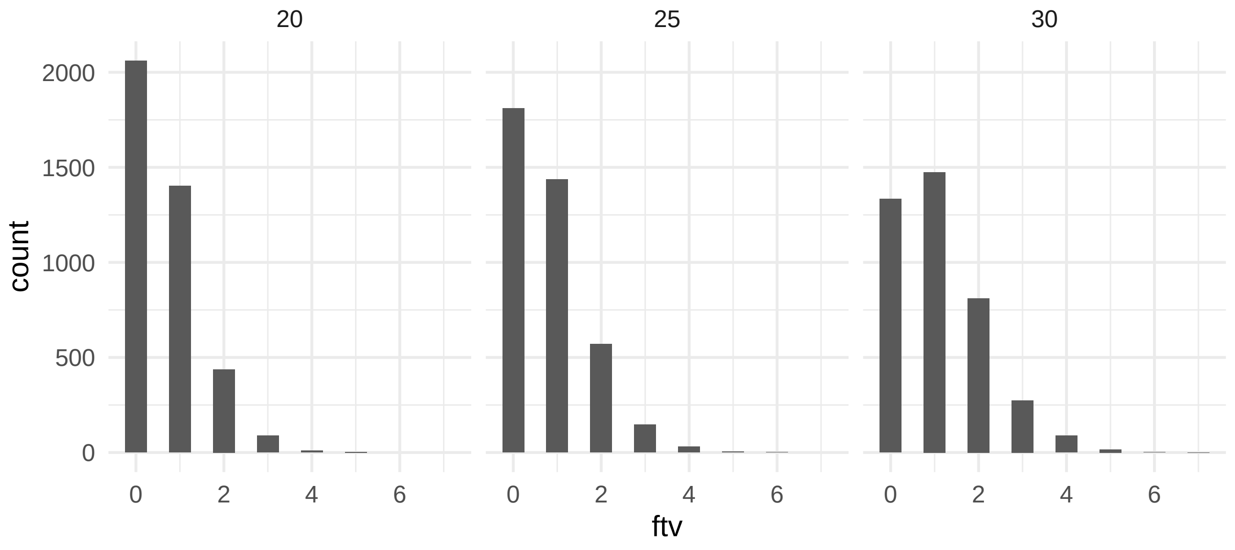


Figure 8: Posterior predictive distribution of the number of visits to the doctor by women of different ages.

References

Bishop, Christopher M. 2006. *Pattern Recognition and Machine Learning*. New York, NY: Springer.

Fair, Ray C. 1978. "A Theory of Extramarital Affairs." *Journal of Political Economy* 86 (1): 45–61.

Murphy, Kevin P. 2012. *Machine Learning: A Probabilistic Perspective.* MIT press.