Chapter 10: Logistic Regression

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 that it can only be applied 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*. In this chapter, we will focus on a class of models known as the logistic regression models. These are some of the most of the widely used examples of the generalized linear models. In the next chapter, we will cover some of the other major types of generalized linear models.

Perhaps the single most common type of logistic regression is binary logistic regression, which we will cover here first. Other types of logistic regression, which are fundamentally related to binary logistic regression include ordinal logistic regression and categorical (or multinomial) logistic regression, and we will also cover these models is this chapter.

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

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

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, which is always either 0 or 1, 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 $1 - \theta$.

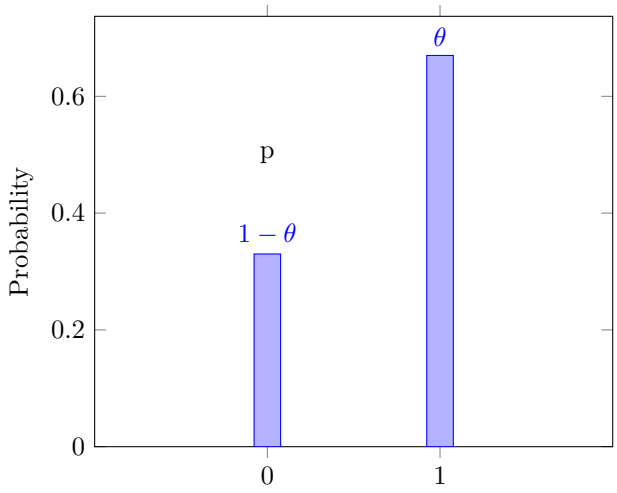

Figure 1: A Bernoulli distribution with parameter θ. The parameter θ gives the probability that the binary variable takes the value of 1. In other words, if x is a binary variable, $\mathrm{P}(x = 1) = \theta$, and so $\mathrm{P}(x = 0) = 1 - \mathrm{P}(x = 1) = 1 - \theta$.

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

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

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 the 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:

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

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

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

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.

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

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

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

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).

```
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.

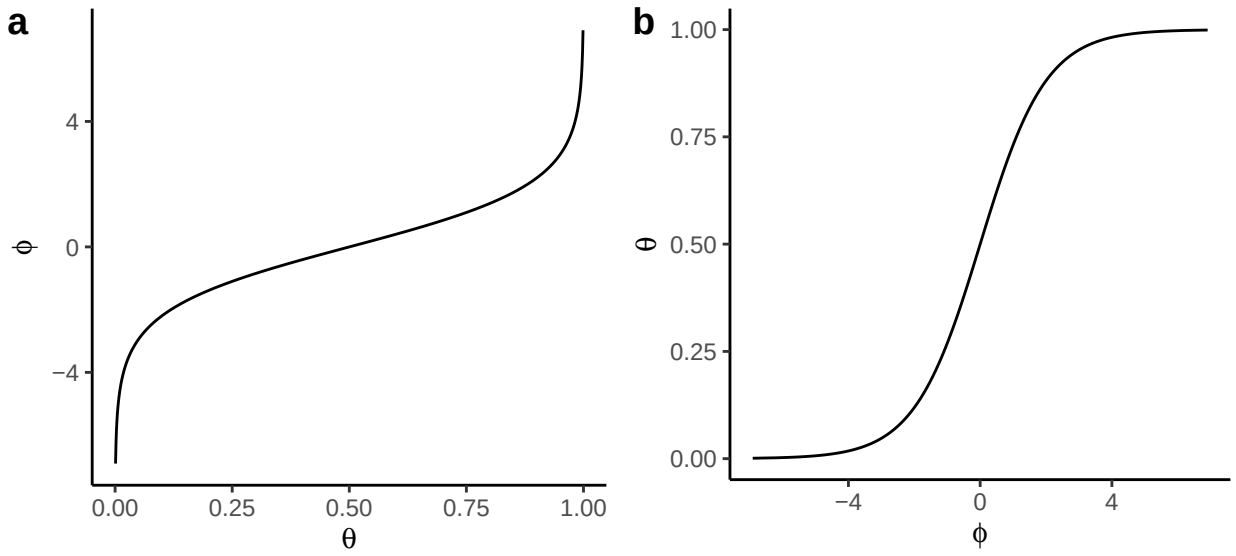

Figure 2: (a) The log odds, also known as the logit, function that maps the interval $(0, 1)$ to $\mathbb{R}$. (b) The inverse of the log odds, also known as the inverse logit, function that maps $\mathbb{R}$ to the interval $(0, 1)$.

```
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 simplify 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.

```
library(magrittr)
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

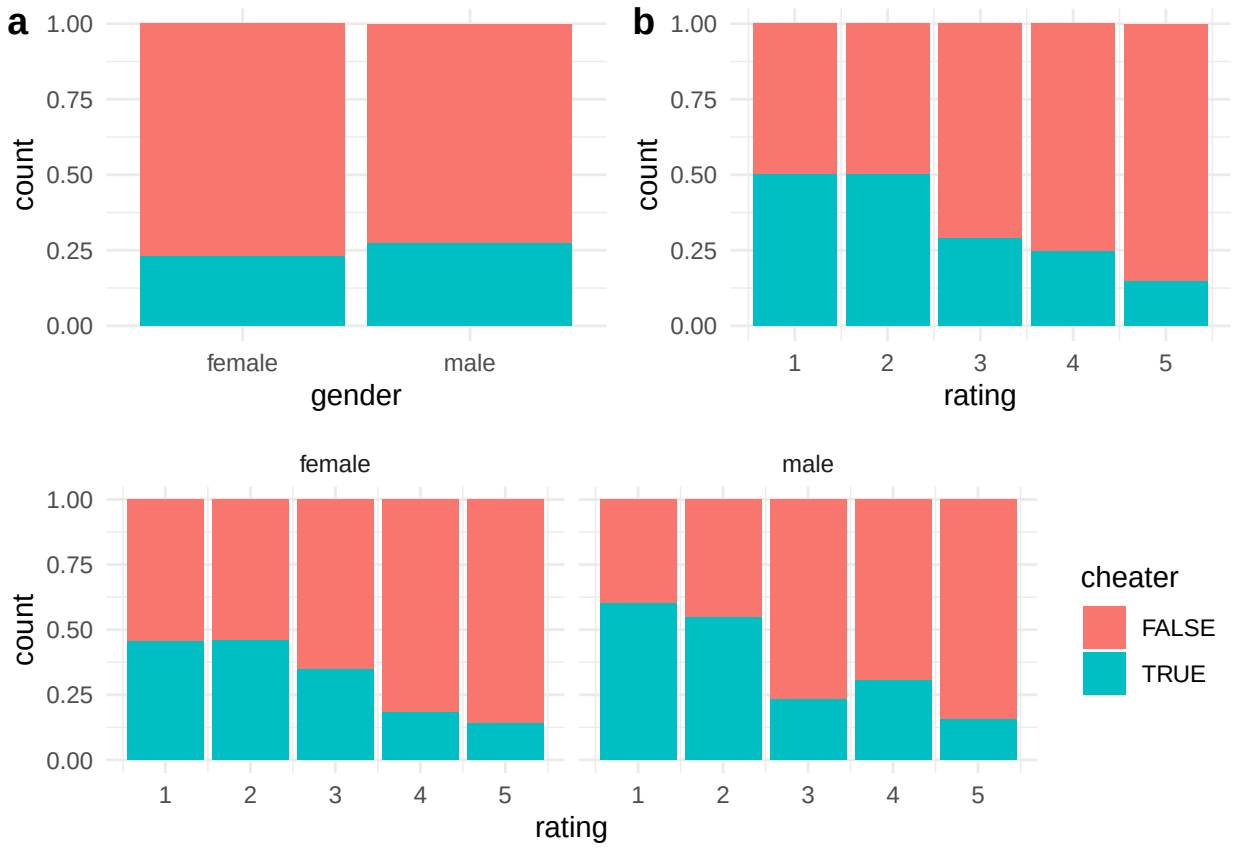

Figure 3: Each bar in each plot shows the proportion of people in the relevant group who have had an affair or not in the past year. (a) The proportions for female and males. (b) The proportions according to the rating of the happiness of the marriage. (c) The proportions according to the marriage rating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

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

or equivalently,

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

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, $\text{ilogit}(\beta_0 + \beta_1 \times 0) = \text{ilogit}(\beta_0)$ will give us the estimate of the probability that a female will have an affair,

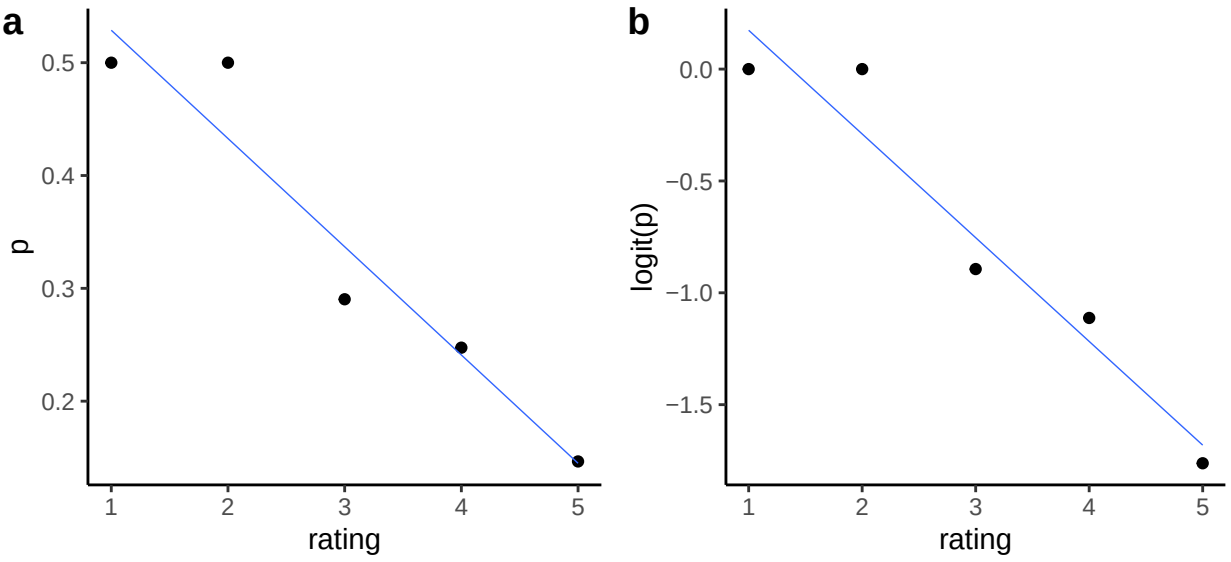

Figure 4: (a) The proportion of cheaters as a function of the marriage rating. (b) The log odds of the proportion of cheaters as a function of the marriage rating.

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

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

which is equivalent to

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

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, 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

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

which is equivalent to

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

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 $\beta_0, \beta_1 \dots \beta_K$ 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 $\beta_0, \beta_1 \dots \beta_K$ 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 $\vec{y} = [y_1, y_2 \dots y_n]^\intercal$, $\vec{\beta} = [\beta_0, \beta_1 \dots \beta_K]^\intercal$, 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, and $\theta_i = \mathrm{ilogit}(X\vec{\beta})$. The logarithm of the likelihood is

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

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, $\log L(\vec{\beta}|\vec{y}, X)$ 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 $t \in 0, 1, 2 \dots$, 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 $\theta_i^t = \mathrm{ilogit}(X\vec{\beta}_t)$, and $\vec{\theta}_t = [\theta_1^t, \theta_2^t \dots \theta_n^t]^\intercal$.

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, $\hat{\text{se}}_k = \sqrt{(X^\intercal S_t X)^{-1}_{kk}}$. 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.

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

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

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

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 `pivot_wider` the predictions for females and males to make them more easy to compare.

```
predictions <- affairs_df_new %>%
  add_predictions(affairs_m, type='response') %>%
  pivot_wider(names_from = gender, values_from = 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

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

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, which is in fact the deviance comparison test for nest model comparison that we saw in Chapter 8. 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 $y_1, y_2 \dots y_i \dots y_n$, where each $y_i \in \{0, 1\}$, and that we have a set of predictors $\vec{x}_1, \vec{x}_2 \dots \vec{x}_i \dots \vec{x}_n$, where each $\vec{x}_i$ is

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

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

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

```
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

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

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 $\vec{\beta}$ 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 described in Chapter 8 and will described further in Chapter 17, 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 1000 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, 2.5) 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 Tail_ESS
```

```
#> Intercept   0.7089189 0.34298420  0.05357066  1.4032220 1.000832     4413     2981
#> gendermale  0.2613167 0.20020218 -0.13226492  0.6528835 1.000461     4097     3401
#> rating     -0.5130190 0.08648511 -0.68262469 -0.3473037 1.000304     3735     2830
```

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))) %>%
  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    -0.314      0.194      0.733
#>  2 female      2    -0.706     -0.315      0.0825
#>  3 female      3    -1.12      -0.829     -0.531
#>  4 female      4    -1.64      -1.34      -1.05
#>  5 female      5    -2.23      -1.85      -1.49
#>  6 male        1    -0.0652     0.452      0.980
#>  7 male        2    -0.443     -0.0564     0.332
#>  8 male        3    -0.855     -0.570     -0.277
#>  9 male        4    -1.35      -1.08      -0.809
#> 10 male        5    -1.94      -1.59      -1.24
```

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

Latent variable formulation

Before we leave binary logistic regression, as we will see when considering other forms of logistic regression, it is useful to consider an alternative formulation of it. In this, y_i is a binary variable that takes the value of 1 or 0 if a latent variable η_i is respectively above or below 0. More precisely, this latent variable formulation is as follows:

$$\text{for } i \in 1 \dots n, \quad y_i = \begin{cases} 1, & \text{if } \eta_i \geq 0, \\ 0, & \text{if } \eta_i < 0 \end{cases},$$
$$\eta_i \sim \text{logistic}(\phi_i, 1).$$

[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.

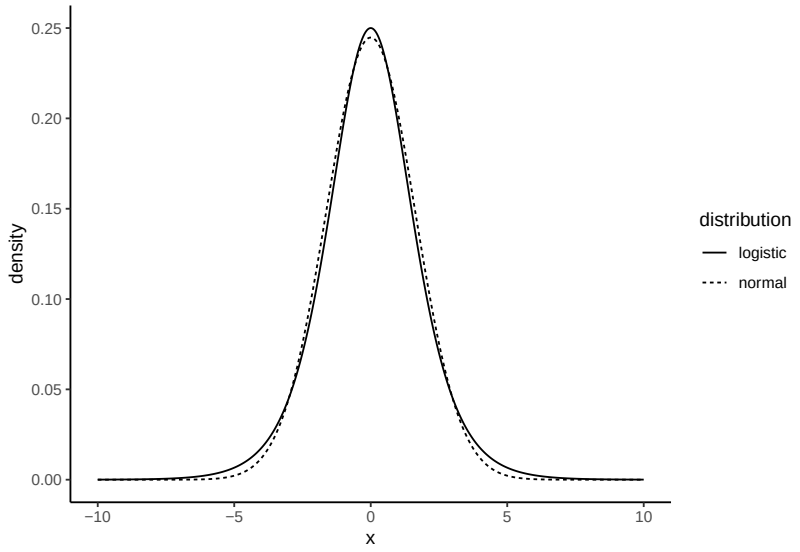

Figure 5: The standard logistic distribution whose mean is equal to 0 and whose scale parameter is 1. This compared with a normal distribution centered at 0 and with standard deviation of 1.63.

Here, η_i is a latent or unobserved random variable that is distributed as a *logistic distribution* whose mean is $\phi_i = \beta_0 + \sum_{k=1}^{K} \beta_k x_{ki}$ and whose scale parameter is equal to 1. The logistic distribution, displayed in Figure 5, is a bell shaped distribution. It is roughly similar to normal distribution centered with a standard deviation of 1.63. The probability density of the logistic distribution with location paramter ϕ_i scale parameter equal to 1 is as follows:

$$\mathrm{P}(\eta_i|\phi_i) = \frac{e^{-(\eta_i - \phi_i)}}{\left(1 + e^{-(\eta_i - \phi_i)}\right)^2}.$$

The cumulative distribution function of the logistic distribution is as follows:

$$\mathrm{P}(\eta_i \leq \omega|\phi_i) = \int_{-\infty}^{\omega} \frac{e^{-(\eta_i - \phi_i)}}{\left(1 + e^{-(\eta_i - \phi_i)}\right)^2} d\eta_i = \frac{1}{1 + e^{-(\omega - \phi_i)}}.$$

From this cumulative distribution function, we can see that the probability that y_i will take the value of 1 is equal to the probability that η_i takes a value of greater than 0, which is as follows:

$$\mathrm{P}(y_i = 1) = 1 - \mathrm{P}(\eta_i < 0|\phi_i) = 1 - \frac{1}{1 + e^{\phi_i}} = \frac{1}{1 + e^{-\phi_i}} = \mathrm{ilogit}(\phi_i).$$

Hence, we see that the latent variable formulation is identical to original definitions of the logistic regression given above, where the probability that y_i takes the value of 1 was equal to θ_i, which was equal to $1/(1+e^{-\phi_i})$.

From this formulation, we see the correspondence between the normal linear model and the binary logistic regression. In the normal linear model, the outcome variable y_i is modelled as a normal distribution whose mean increases or decreases as a linear function of a set of predictors. In the binary logistic regression, a latent variable η_i is modelled as logistic distribution whose mean increases or decreases as a linear function of a set of predictors, and the binary outcome variable y_i takes the value of 0 or 1 depending on whether this latent variable is, respectively, above or below 0. We can see that the latent variable as an evidence variable, and so the distribution over it as a distribution over the evidence in favour of one outcome value $y_i = 1$ or another $y_i = 0$. In Figure 6, we show three logistic distributions over a latent variable η. We can view these as three different representations of the degree of evidence for the value of the outcome variable.

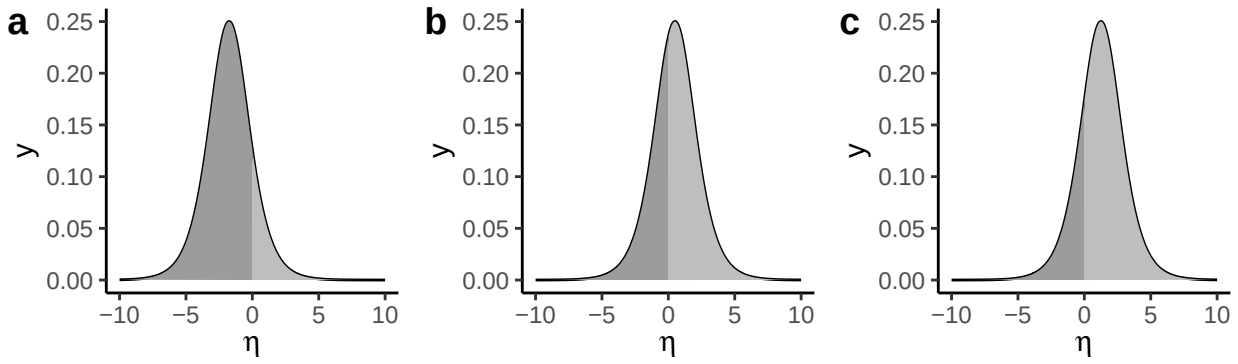

Figure 6: Three logistic distributions with means a) -1.75 b) 0.50 c) 1.25. Each distribution can seen as a representation of the degree of evidence of the outcome variable y_i taking the value of 1. As the distribution shifts to the right, there is more evidence in favour of the outcome variable taking the value of 1. In these three distributions, the probabilities that the outcome is 1 corresponds to 0.15, 0.62, and 0.78.

Probit regression

Having seen the latent variable formulation of binary logistic regression, we are able to more easily understand *probit regression*, which is a regression model that is very closely related to binary logistic regression regression. In probit regression, each outcome variable y_i is binary. They are modelled as follows:

$$\text{for } i \in 1 \dots n, \quad y_i = \begin{cases} 1, & \text{if } \eta_i \geq 0, \\ 0, & \text{if } \eta_i < 0 \end{cases} \quad ,$$
$$\eta_i \sim \mathrm{N}(\phi_i, 1),$$

where $N(\phi_i, 1)$ is a normal distribution with mean of ϕ_i and standard deviation of 1, and where ϕ_i is a linear function of a set of predictors, just as above. By direct analogy with the latent variable formulation of the binary logistic model, in the probit model, the probability that y_i will take the value of 1 is equal to the probability that η_i takes a value of greater than 0, which is as follows:

$$\mathrm{P}(y_i = 1) = 1 - \mathrm{P}(\eta_i < 0 | \phi_i) = 1 - \Phi(-\phi_i) = \Phi(\phi_i),$$

where Φ is the cumulative distribution function in a standard normal distribution. In order words, in binary logistic regression, we have

$$\mathrm{P}(y_i = 1) = \mathrm{ilogit}(\phi_i),$$

while in probit regression, we have

$$\mathrm{P}(y_i = 1) = \Phi(\phi_i).$$

Thus, binary logistic regression and probit differ by their *link* function. It is the log odds or logit function in the case of logistic regression, and it is the *quantile function* Φ^{-1}, which is the inverse of the cumulative distribution function Φ, of the standard normal in the case of probit regression. These functions are shown in Figure 7.

We can perform a probit regression in R using `glm` just like in the case of binary logistic regression, but using `link = 'probit'` instead of `link = 'logit'`. In the following, using the `affairs_df` data, we model how the probability of being a cheater varies as a function of the `rating` variable.

```
affairs_probit <- glm(cheater ~ rating,
                      family = binomial(link = 'probit'),
                      data = affairs_df)
```

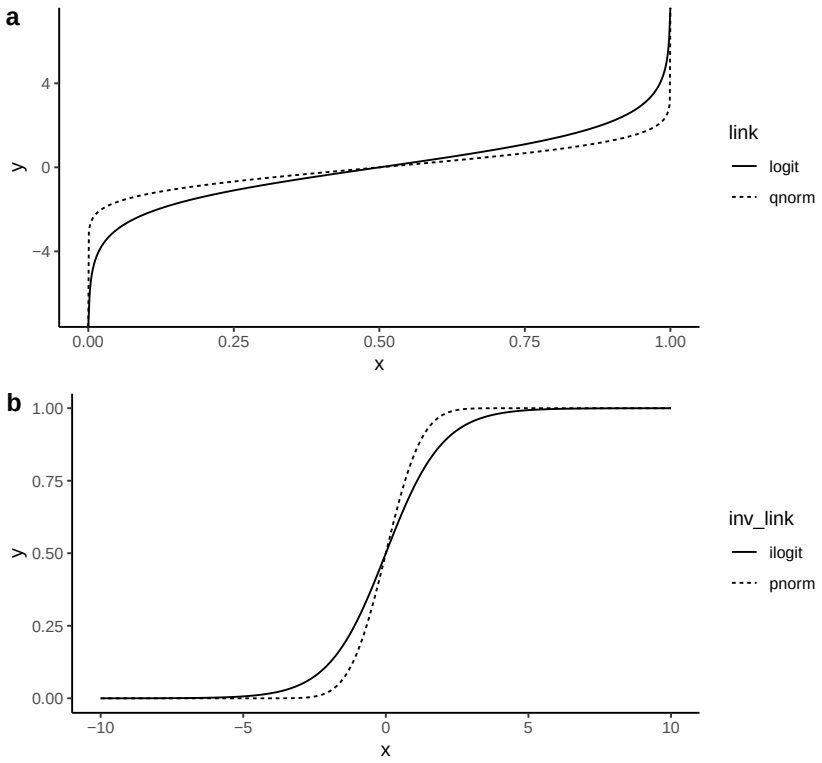

Figure 7: a) The logit and quantile function of the normal distribution, which are the link functions of binary logistic regression and probit regression, respectively. b) The inverse logit and cumulative distribution function of the normal distribution.

In probit regression, statistical inference is identical to binary logistic regression: the maximum likelihood estimator of the regression coefficients is found by iteratively reweighted least squares, whose sampling distribution is asymptotically normal. The coefficients summary table for `affairs_probit` are as follows.

```
summary(affairs_probit)$coefficients
#>               Estimate Std. Error   z value     Pr(>|z|)
#> (Intercept)  0.4817451 0.19788927  2.434418 1.491577e-02
#> rating      -0.3031677 0.05023335 -6.035187 1.587786e-09
```

Comparing this to the set of coefficients for the binary logistic regression, we see that are considerable differences in the scale of the coefficients and their standard errors, though with the z test statistics being very similar.

```
glm(cheater ~ rating, family = binomial(link = 'logit'), data = affairs_df) %>%
  summary() %>%
  extract2('coefficients')
#>               Estimate Std. Error   z value     Pr(>|z|)
#> (Intercept)  0.8253902 0.32548132  2.535907 1.121566e-02
#> rating      -0.5082193 0.08468845 -6.001046 1.960510e-09
```

Importantly, in the probit regression, we do not interpret the coefficients in terms of odds ratios. In probit regression, the coefficient gives the change in mean of the (unit variance) normal distribution over the latent variable in the probit model for every unit change in the predictor. For example, in the `affairs_probit` model, the coefficient for `rating` is -0.303. This means that as `rating` increases by one unit, the mean of the normal distribution over the latent variable increases by -0.303 unit.

Prediction in probit regression works like prediction in binary logistic regression, but we apply the inverse

of the link function, which is the standard normal, to convert from the value of the linear predictor ϕ_i to the probability that $y_i = 1$. We can do this using the `predict` and `add_predictions` function using `type = 'response'` as follows.

```
tibble(rating = seq(5)) %>%
  add_predictions(affairs_probit, type = 'response')
#> # A tibble: 5 x 2
#>   rating  pred
#>    <int> <dbl>
#> 1      1 0.571
#> 2      2 0.450
#> 3      3 0.334
#> 4      4 0.232
#> 5      5 0.151
```

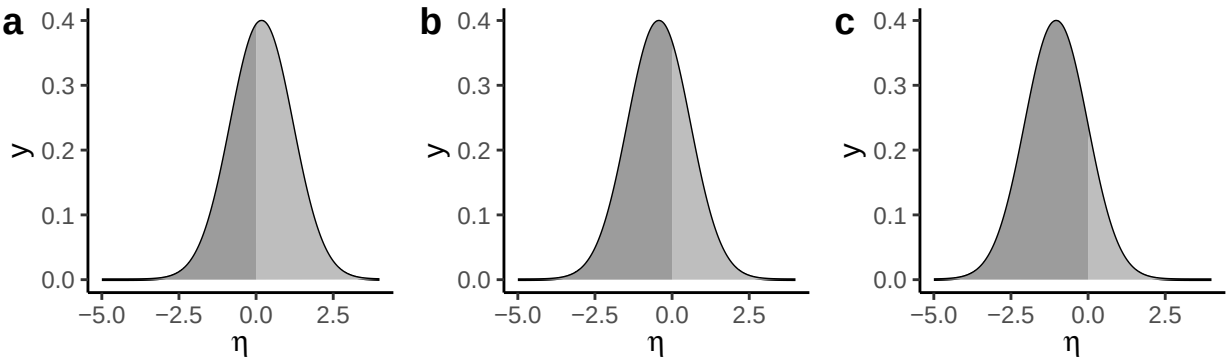

Figure 8: The distribution over the latent variable in the probit regression model that models the probability of being a cheater as a function of `rating`, for values of `rating` equal to a) 1 b) 3, and c) 5. When `rating` increases by one unit, the mean of the normal distribution increases by -0.30, which is the value of the coefficient for `rating`. The areas shaded to the right are the corresponding probabilities that the outcome variable y_i takes the value of 1, which in this case means the probability that the person has had an extra-martial affair. These probabilities are 0.57, 0.33, and 0.15, respectively.

Ordinal logistic regression

In ordinal regression, the outcome variable y_i is an ordinal variable. An ordinal variable can be seen as a categorical variable where the values have an order. Alternatively, an ordinal variable can be seen as a numerical variable whose values can be ordered but not defined on a metric space. For example, the values of y_i could be *low*, *medium*, *high*. Here, there is a natural order: *low* < *medium* < *high*. However, we do not necessarily believe that difference between *low* and *medium* is the same as between *medium* and *high*. We may represent the *low*, *medium*, *high* by $\{0, 1, 2\}$. We treat these values as essentially labels, just as we would do with a categorical variable, though with the understanding that $0 < 1 < 2$.

In a regression model with an ordinal outcome variable, we model the probability distribution over the outcome variable and how it changes with a set of predictor variables. For example, consider the *world values surveys* data that is available in `carData` by the name `WVS`.

```
library(carData)
wvs_df <- as_tibble(WVS)
wvs_df
#> # A tibble: 5,381 x 6
#>    poverty     religion degree country   age gender
```

```
#>    <ord>       <fct>    <fct>  <fct>   <int> <fct>
#>  1 Too Little  yes      no     USA        44 male
#>  2 About Right yes      no     USA        40 female
#>  3 Too Little  yes      no     USA        36 female
#>  4 Too Much    yes      yes    USA        25 female
#>  5 Too Little  yes      yes    USA        39 male
#>  6 About Right yes      no     USA        80 female
#>  7 Too Much    yes      no     USA        48 female
#>  8 Too Little  yes      no     USA        32 male
#>  9 Too Little  yes      no     USA        74 female
#> 10 Too Little  yes      no     USA        30 male
#> # ... with 5,371 more rows
```

In this data set, we have a variable `poverty` that represents the responses to the survey question *Do you think that what the government is doing for people in poverty in this country is about the right amount, too much, or too little?*. This variable takes the values of `Too Little`, `About Right`, and `Too much`. Note that this variable is an ordered factor.

```
wvs_df %>% pull(poverty) %>% class()
#> [1] "ordered" "factor"
```

In other words, it is a factor variable whose levels have a defined order, namely the following.

```
wvs_df %>% pull(poverty) %>% levels()
#> [1] "Too Little"  "About Right" "Too Much"
```

In addition to `poverty`, we have predictor variables such as `religion`, `gender`, `age`, etc. In Figure 9, we group the `age` variable into 5 quintiles, and plot the numbers of males and female respondents in these quintiles who choose each of the three responses to the `poverty` question. From this data, we can see that as age increases, the number of people responding `Too Little` declines, and the numbers of people responding either `About Right` or `Too Much` increase. We also see that more females than males respond `Too Little`, and usually more males than females respond `About Right` or `Too Much`.

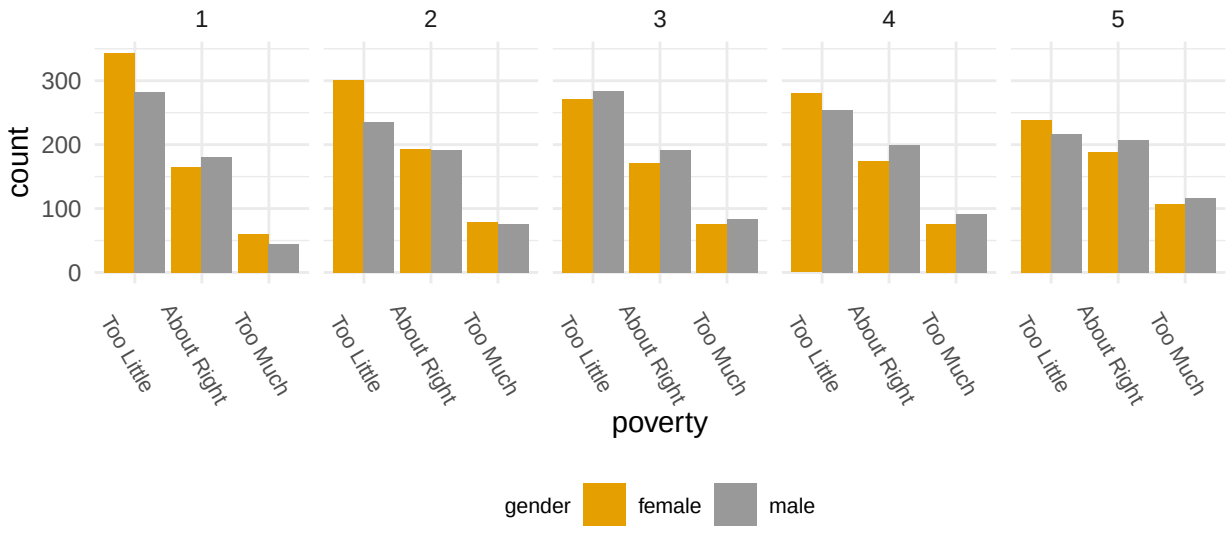

Figure 9: The frequencies of choosing each possible response to a survey question about what the government is doing about poverty, as a function of the respondent's age quintile (lower means younger) and gender.

One of the most widely used regression models for ordinal outcome data is the *proportional odds* or *cumulative*

logit logistic regression model. In the case of an ordinal outcome variable with three values, i.e., $y_i \in \{1, 2, 3\}$, this model is equivalent to the latent variable formulation of the binary logistic regression that we saw above, but with two rather than one thresholds, which are known as *cutpoints*, and denoted here by ζ_1 and ζ_2. In particular, the model is as follows.

$$\text{for } i \in 1 \dots n, \quad y_i = \begin{cases} 3, & \text{if } \eta_i \geq \zeta_2, \\ 2, & \text{if } \zeta_1 \leq \eta_i < \zeta_2, \\ 1, & \text{if } \eta_i < \zeta_1 \end{cases},$$
$$\eta_i \sim \text{logistic}(\phi_i, 1).$$

From this we have,

$$\text{P}(y_i \leq 2) = \int_{\infty}^{\zeta_1} \text{P}(\eta_i|\phi_i) d\eta_i = \frac{1}{1 + e^{-(\zeta_2 - \phi_i)}} = \text{ilogit}(\zeta_2 - \phi_i),$$
$$\text{P}(y_i \leq 1) = \int_{\infty}^{\zeta_0} \text{P}(\eta_i|\phi_i) d\eta_i = \frac{1}{1 + e^{-(\zeta_1 - \phi_i)}} = \text{ilogit}(\zeta_1 - \phi_i),$$

and so $\text{P}(y_i = 3) = 1 - \text{P}(y_i \leq 2) = 1 - \text{ilogit}(\zeta_2 - \phi_i)$. Stating these cumulative probabilities in terms of log odds, we have

$$\log \left(\frac{\text{P}(y_i \leq 2)}{1 - \text{P}(y_i \leq 2)} \right) = \zeta_2 - \phi_i,$$
$$\log \left(\frac{\text{P}(y_i \leq 1)}{1 - \text{P}(y_i \leq 1)} \right) = \zeta_1 - \phi_i.$$

In general, for an ordinal variable with J levels, $1 \dots J$, for $j \in 1 \dots J - 1$, we have

$$\log \left(\frac{\text{P}(y_i \leq j)}{1 - \text{P}(y_i \leq j)} \right) = \zeta_j - \phi_i,$$

where $\zeta_1 < \zeta_2 < \ldots < \zeta_{J-1}$.

The value of ϕ_i is, as it was used above, the linear sum of the predictors, i.e. $\phi_i = \beta_0 + \sum_{k=1}^{K} \beta_k x_{ki}$. Having the intercept term β_0 as well the cutpoints $\zeta_1, \zeta_2 \dots \zeta_{J-1}$ means that neither are identifiable because we could add any constant value to β_0 and subtract this value from each of $\zeta_1, \zeta_2 \dots \zeta_{J-1}$ to obtain an identical model. For this reason, we must constrain the values of either β_0 or the cutpoints. One possibility is to contrain ζ_1 to equal 0. Another, which is more common, is to contrain β_0 to equal to 0. In other words, $\phi_i = \sum_{k=1}^{K} \beta_k x_{ki}$.

The general model for the cumulative logits ordinal logistic regression is therefore the following.

$$\text{for } i \in 1 \dots n, \quad y_i = \begin{cases} J, & \text{if } \eta_i \geq \zeta_{J-1}, \\ J-1, & \text{if } \zeta_{J-2} \leq \eta_i < \zeta_{J-1}, \\ \dots, & \\ 2, & \text{if } \zeta_1 \leq \eta_i < \zeta_2, \\ 1, & \text{if } \eta_i < \zeta_1 \end{cases},$$
$$\eta_i \sim \text{logistic}(\phi_i, 1), \quad \phi_i = \sum_{k=1}^{K} \beta_k x_{ki}.$$

For any $1 \leq j \leq J - 1$, we have

$$\begin{aligned} \text{P}(y_i = j) &= \text{P}(y_i \leq j) - \text{P}(y_i \leq j - 1), \\ &= \text{ilogit}(\zeta_j - \phi_i) - \text{ilogit}(\zeta_{j-1} - \phi_i). \end{aligned}$$

Ordinal logistic regression in R

There are many options to perform ordinal logistic regression in R. For example, the `ordinal` package is an excellent package for many variants of the ordinal logistic model including and especially mixed effect ordinal models. Here, however, we will use the `polr` function from the `MASS` package which is simple and easy to use and perfectly illustrates ordinal logistic regression as we have described it thus far. We will use it here to model how the `poverty` variable varies as a function of `age` and `gender` in the `wvs_df` data set.

```
library(MASS)
M_ord <- polr(poverty ~ age + gender, data = wvs_df)
summary(M_ord)
#> Call:
#> polr(formula = poverty ~ age + gender, data = wvs_df)
#>
#> Coefficients:
#>               Value Std. Error t value
#> age         0.01308   0.001523   8.592
#> gendermale 0.15411   0.052139   2.956
#>
#> Intercepts:
#>                           Value   Std. Error t value
#> Too Little|About Right  0.6762  0.0779     8.6794
#> About Right|Too Much    2.4123  0.0850    28.3761
#>
#> Residual Deviance: 10656.41
#> AIC: 10664.41
```

In the summary, the values of the coefficients for the linear sum are listed under `Coefficients`, while the values of ζ_1 and ζ_2 are listed under `Intercepts`. The maximum likelihood estimates of the coefficients and the cutpoints are calculated using a general purpose optimization based on the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm, and estimates the standard error of these estimates using the Hessian matrix of the log-likelihood function evaluated at its maximum. Note that the summary output does not contain p-values for either the coefficients or for the cutpoints. The authors of MASS (Venables and Ripley 2002) state that the exact distribution of the estimates are not known and so exact p-values can not be calculated. Alternative methods for evaluating coefficients, based on likelihood ratio tests, are recommended instead. However, as an approximation, though one to be used cautiously, we can treat the sampling distribution for these estimates as normally distributed and so treat the `t value` as a standard normal statistic and calculate the p-value using the cumulative normal distribution.

Although, as we will see, we can use R's generic `predict` function to calculate the probabilities for the `poverty` outcome variable for any given set of values for the predictor variables, it is instructive to do this manually. For example, for a male of median age (for men), which is 44 years in this data set, the value of ϕ_i is

$$\begin{aligned}\phi_i &= 0.0131 \times 44 + 0.1541 \times 1, \\ &= 0.73.\end{aligned}$$

The log odds that this median aged male responds that `poverty` is "Too Little" and `poverty` is "Too Little" or "About Right" are, respectively, $\zeta_1 - \phi_i = 0.676 - 0.73 = -0.05$ and $\zeta_2 - \phi_i = 2.412 - 0.73 = 1.68$. These correspond to cumulative probabilities of $\text{ilogit}(\zeta_1 - \phi_i) = 0.49$ and $\text{ilogit}(\zeta_2 - \phi_i) = 0.84$. Thus, the probability that the median aged male responds by "Too Much", "About Right", or "Too Little" is, respectively, $1 - \text{ilogit}(\zeta_2 - \phi_i) = 0.16$, $\text{ilogit}(\zeta_2 - \phi_i) - \text{ilogit}(\zeta_1 - \phi_i) = 0.36$, and $\text{ilogit}(\zeta_1 - \phi_i) = 0.49$. The logistic distribution corresponding to $\phi = 0.73$ is shown in Figure 10.

Using the `predict` or `add_predictions` functions, we can easily calculate the probabilities over the outcome variable's values for any given set of values of the predictor variables. To do so, we must use the `type = 'probs'` argument when calling these functions.

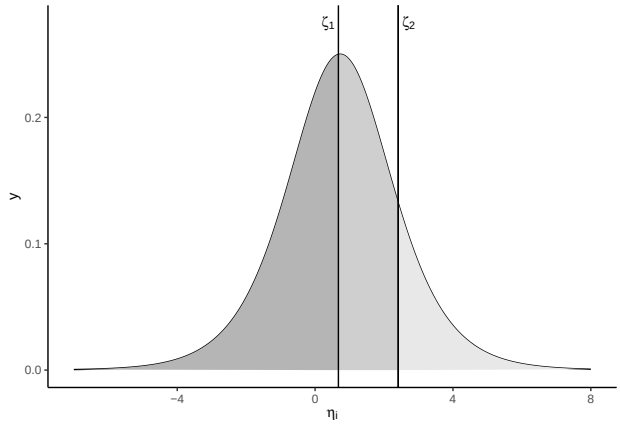

Figure 10: The partitioning of the area under the logistic distribution whose mean is $\phi_i = 0.73$, which is the value of the linear predictor for a median aged male. The two cutpoints are $\zeta_1 = 0.68$ and $\zeta_2 = 2.41$.

```
new_data <- expand_grid(age = c(25, 50, 75),
                        gender = c('male', 'female'))
add_predictions(new_data, M_ord, type='probs')
#> # A tibble: 6 x 3
#>     age gender pred[,"Too Little"] [,"About Right"] [,"Too Much"]
#>   <dbl> <chr>                <dbl>            <dbl>         <dbl>
#> 1    25 male                 0.549            0.325         0.127
#> 2    25 female               0.586            0.303         0.111
#> 3    50 male                 0.467            0.366         0.167
#> 4    50 female               0.505            0.347         0.147
#> 5    75 male                 0.387            0.395         0.218
#> 6    75 female               0.424            0.383         0.193
```

Odds ratios

We can interpret the values of the cutpoints and the regression coefficients in terms of odds ratios. We saw above that, for all $j \in 1 \dots J$,

$$\log\left(\frac{\mathrm{P}(y_i \leq j)}{1 - \mathrm{P}(y_i \leq j)}\right) = \zeta_j - \phi_i.$$

From this, for any two cutpoints $j' > j$, we have

$$\begin{aligned}
\log\left(\frac{\mathrm{P}(y_i \leq j')}{1 - \mathrm{P}(y_i \leq j')}\right) - \log\left(\frac{\mathrm{P}(y_i \leq j)}{1 - \mathrm{P}(y_i \leq j)}\right) &= (\zeta'_j - \phi_i) - (\zeta_j - \phi_i), \\
\log\left(\frac{\mathrm{P}(y_i \leq j')}{1 - \mathrm{P}(y_i \leq j')} \Big/ \frac{\mathrm{P}(y_i \leq j)}{1 - \mathrm{P}(y_i \leq j)}\right) &= \zeta'_j - \zeta_j, \\
\frac{\mathrm{P}(y_i \leq j')}{1 - \mathrm{P}(y_i \leq j')} \Big/ \frac{\mathrm{P}(y_i \leq j)}{1 - \mathrm{P}(y_i \leq j)} &= e^{\zeta'_j - \zeta_j}.
\end{aligned}$$

From this, we see that $e^{j'-j}$ is the odds ratio corresponding to the probabilities $\mathrm{P}(y_i < j')$ and $\mathrm{P}(y_i < j)$. In other words, the ratio of the odds that $y_i < j'$ to the odds that $y_i < j'$, for any j' and j, is $e^{j'-j}$. Note that this will hold for any value of ϕ and so holds for any set of values of the predictors.

To interpret the coefficients, consider increasing the value of any predictor k by one unit. If $\phi_i = \sum_{k=1}^{K} \beta_k x_{ki}$, if we increase x_{ki} by one unit, we have $\phi'_i = \sum_{k=1}^{K} \beta_k x_{ki} + \beta_k = \phi + \beta_k$. For any value j of the ordinal

outcome variable, we have

$$\log\left(\frac{\mathrm{P}(y_i \leq j|\phi')}{1-\mathrm{P}(y_i \leq j|\phi')}\right) - \log\left(\frac{\mathrm{P}(y_i \leq j|\phi)}{1-\mathrm{P}(y_i \leq j|\phi)}\right) = (\zeta_j - \phi_i') - (\zeta_j - \phi_i) = \beta_k,$$
$$\frac{\mathrm{P}(y_i \leq j|\phi')}{1-\mathrm{P}(y_i \leq j|\phi')} \Big/ \frac{\mathrm{P}(y_i \leq j|\phi)}{1-\mathrm{P}(y_i \leq j|\phi)} = e^{\beta_k}.$$

In other words, e^{β_k} is the factor by which the odds that $y_i \leq j$, for any $j \in 1 \ldots J-1$, increases for every one unit increase in predictor variable k.

Bayesian ordinal logistic regression

We can perform a Bayesian counterpart of the cumulative logit ordinal logistic regression model as follows.

```
M_ord_bayes <- brm(poverty ~ age + gender,
                   family = cumulative(link = 'logit'),
                   data = wvs_df)
```

As we can see, all we need specify is that the family is `cumulative` and the link is `logit`, which is the default in fact. By using all the other default settings, we use 4 chains, each with 2000 iterations, and where the initial 1000 iterations are discarded, leaving to 4000 total samples.

The default priors in this model are seen in the following table.

```
prior_summary(M_ord_bayes)
#>                   prior     class       coef group resp dpar nlpar bound
#> 1                               b
#> 2                               b        age
#> 3                               b gendermale
#> 4 student_t(3, 0, 2.5) Intercept
#> 5                      Intercept          1
#> 6                      Intercept          2
```

This tells us that the cutpoints have non-standard student t-distributions and the coefficients have uniform distributions as priors.

The summary of the posterior distribution of the regression coefficients and the cutpoints are as follows:

```
summary(M_ord_bayes)$fixed
#>                Estimate   Est.Error   l-95% CI   u-95% CI      Rhat Bulk_ESS Tail_ESS
#> Intercept[1] 0.67727898 0.079030844 0.51950236 0.83619957 0.9996931     4446     2906
#> Intercept[2] 2.41520856 0.086665885 2.24881906 2.58807770 1.0001916     4349     2911
#> age          0.01311032 0.001544013 0.01007376 0.01612692 0.9995239     4880     2799
#> gendermale   0.15377660 0.050522945 0.05841022 0.25147234 1.0018276     3474     2918
```

Clearly, the means of these estimates are very similar to those estimated using maximum likelihood estimation above.

Categorical (multinomial) logistic regression

Thus far, we have considered regression models where the outcome variable is either a binary variable or an ordinal variable. If the outcome variable is a categorical variable with more than two values, we can use an extension of logistic regression that we will refer to here as categorical logistic regression, but which is more also commonly referred to as multinomial logistic regression. We prefer the term categorical, rather than multinomial, logistic regression for this model given that the outcome variable is a categorical variable. We prefer to reserve the term multinomial logistic regression for case of models where the outcome variable is vector of counts of the number of observations of each of a set of categorically distinct outcomes.

In categorical logistic regression, for each observation, our outcome variable can be represented as $y_i \in 1 \dots J$, where $1 \dots J$ are J categorically distinct values. For example, in a hypothetical pre-election voting preference poll in the UK, people might be asked if they will vote Conservative, Labour, Liberal Democrats, Green, Other. We might then model how the probability distribution over these choice vary by UK region, age, gender, etc.

In categorical logistic regression, we model the log of the probability of each category relative to an arbitrarily chosen baseline category as a linear function of the predictors. Setting the baseline category to be $j = 1$, then for each $j = 2 \dots J$, we have

$$\log\left(\frac{\mathrm{P}(y_i = j)}{\mathrm{P}(y_i = 1)}\right) = \phi_{ji} = \beta_{j0} + \sum_{k=1}^{K} \beta_{jk} x_{ki},$$

and by necessity, we have

$$\log\left(\frac{\mathrm{P}(y_i = 1)}{\mathrm{P}(y_i = 1)}\right) = \phi_{1i} = 0.$$

In other words, we model the log of the probability that $y_i = j$ relative to $y_i = 1$ as a linear function of the predictors. Note that we have a separate linear model with different coefficients for each $j > 2$. We can interpret the log ratio

$$\log\left(\frac{\mathrm{P}(y_i = j)}{\mathrm{P}(y_i = 1)}\right) = \phi_{ji}$$

in one of two ways. We can either interpret it directly as simply the log of a relative probabilities. Alternatively, we can interpret it as the log odds of the conditional probability that $y_i = j$ given that we know that $y_i = j$ or $y_i = 1$.

From this model, we have

$$\mathrm{P}(y_i = j) = e^{\phi_{ji}} \mathrm{P}(y_i = 1),$$

and given that, by definition, we have

$$\sum_{j=1}^{J} \mathrm{P}(y_i = j) = 1,$$

we therefore have the following:

$$\mathrm{P}(y_i = 1) \sum_{j=1}^{J} e^{\phi_{ji}} = 1$$

$$\mathrm{P}(y_i = 1) = \frac{1}{\sum_{j=1}^{J} e^{\phi_{ji}}}.$$

This leads to the following model of the probabilities of each of the J values of the outcome variable.

$$\mathrm{P}(y_i = j) = \frac{e^{\phi_{ji}}}{\sum_{j=1}^{J} e^{\phi_{ji}}}.$$

Given that $e^{\phi_{1i}} = 1$, it is more common to write this as

$$\mathrm{P}(y_i = j) = \frac{e^{\phi_{ji}}}{1 + \sum_{j=2}^{J} e^{\phi_{ji}}}.$$

Categorical logistic regression using R

We have many options for doing categorical logistic regression in R. One simple option is to use `multinom` from the `nnet` package.

To illustrate this model, we will use a data set based on a subset of the `weather_check` data set in the `fivethirtyeight` package.

```
weather_df <- read_csv('data/weather.csv')
weather_df
#> # A tibble: 916 x 2
#>    weather         age
#>    <chr>           <chr>
#>  1 app             30 - 44
#>  2 app             18 - 29
#>  3 app             30 - 44
#>  4 app             30 - 44
#>  5 app             30 - 44
#>  6 app             18 - 29
#>  7 weather_channel 30 - 44
#>  8 weather_channel 30 - 44
#>  9 app             30 - 44
#> 10 internet        18 - 29
#> # ... with 906 more rows
```

In this data, people were asked what was their source of information about the weather (`weather`). This had values `app` for a mobile device app, `internet` for general internet search, `tv` for local television, `weather_channel` for the weather channel, and `other` for other sources like newspaper, newsletter, etc. The respondents' ages were listed as the age groups `18 - 29`, `30 - 44`, `45 - 59`, `60+`. The frequency of each response for each age group is as follows:

```
weather_df %>%
  group_by(age, weather) %>%
  tally() %>%
  pivot_wider(id_cols = age, names_from = 'weather', values_from = n)
#> # A tibble: 4 x 6
#> # Groups:   age [4]
#>   age       app internet other    tv weather_channel
#>   <chr>   <int>    <int> <int> <int>           <int>
#> 1 18 - 29    92       35     9    14              26
#> 2 30 - 44   108       26     8    34              28
#> 3 45 - 59   108       33    21    67              49
#> 4 60+        80       36    33    74              35
```

From this, we see a relative increase with age for television, particularly local television, and a relative decline with age for mobile apps.

The following code models the probability distribution of the different weather news sources. First, we will set the `age` and `source` variables as factors, which will order the results to make them easier to interpret.

```
weather_df %<>%
  mutate(age = factor(age, levels = c('18 - 29', '30 - 44', '45 - 59', '60+')),
         weather = factor(weather, levels = c('other', 'app', 'internet', 'tv', 'weather_channel'))
  )
```

For simplicity, we will begin with a model that has a single constant term.

```
M_cat <- multinom(weather ~ 1, data = weather_df)
```

The values of coefficients are estimated using a BFGS based optimization of the log of the likelihood, as was also done above in the case of ordinal logistic regression. Note that `weather == 'other'` is the baseline against which all other weather news sources are compared.

```
summary(M_cat)
#> Call:
#> multinom(formula = weather ~ 1, data = weather_df)
```

```
#>
#> Coefficients:
#>                 (Intercept)
#> app               1.6981849
#> internet          0.6047535
#> tv                0.9789591
#> weather_channel   0.6644412
#>
#> Std. Errors:
#>                 (Intercept)
#> app               0.1290746
#> internet          0.1475638
#> tv                0.1391899
#> weather_channel   0.1460458
#>
#> Residual Deviance: 2656.368
#> AIC: 2664.368
```

In order to appreciate the meaning of the coefficients, it helps to calculate the probability distribution over the five options using the formula

$$\mathrm{P}(y_i = j) = \frac{e^{\phi_{ji}}}{1 + \sum_{j=2}^{J} e^{\phi_{ji}}}.$$

For $j \in 2, 3, 4, 5$, for all i, $\phi_{ji} = \beta_j$, while $\phi_{1i} = 0$.

```
phi <- rbind(other = 0, coef(M_cat)) %>% as_tibble(rownames = 'id') %>% deframe()
phi
#>           other             app        internet              tv weather_channel
#>       0.0000000       1.6981849       0.6047535       0.9789591       0.6644412
```

From this, the corresponding probabilities are as follows:

```
exp(phi)/sum(exp(phi))
#>           other             app        internet              tv weather_channel
#>      0.07751992      0.42357044      0.14192354      0.20633355      0.15065255
```

We can now use `age` as a predictor as follows.

```
M_cat_2 <- multinom(weather ~ age, data = weather_df)
```

```
summary(M_cat_2)
#> Call:
#> multinom(formula = weather ~ age, data = weather_df)
#>
#> Coefficients:
#>                 (Intercept) age30 - 44 age45 - 59     age60+
#> app               2.3244935  0.2782843 -0.6867980 -1.4389520
#> internet          1.3580627 -0.1793077 -0.9059785 -1.2710166
#> tv                0.4417131  1.0052996  0.7185451  0.3658706
#> weather_channel   1.0607799  0.1920954 -0.2133944 -1.0019129
#>
#> Std. Errors:
#>                 (Intercept) age30 - 44 age45 - 59    age60+
#> app               0.3492458  0.5062047  0.4229123 0.4059266
#> internet          0.3737301  0.5505876  0.4664758 0.4446970
#> tv                0.4272412  0.5804809  0.4950589 0.4757649
#> weather_channel   0.3867369  0.5570370  0.4664723 0.4565528
#>
```

```
#> Residual Deviance: 2589.321
#> AIC: 2621.321
```

To calculate the probability distributions over `weather`, rather than doing so manually, we can use `predict` or `add_predictions` with `type = 'probs'`.

```
tibble(age = c('18 - 29', '30 - 44', '45 - 59', '60+')) %>%
  add_predictions(M_cat_2, type='probs')
#> # A tibble: 4 x 2
#>   age     pred[,"other"] [,"app"] [,"internet"] [,"tv"] [,"weather_channel"]
#>   <chr>            <dbl>    <dbl>         <dbl>   <dbl>                <dbl>
#> 1 18 - 29         0.0511    0.523         0.199  0.0795                0.148
#> 2 30 - 44         0.0392    0.529         0.127  0.167                 0.137
#> 3 45 - 59         0.0755    0.388         0.119  0.241                 0.176
#> 4 60+             0.128     0.310         0.140  0.287                 0.136
```

These predicted probabilities are shown in Figure 11. What is most clear here is that we see that mobile device apps decline, and local television increases, as age increases.

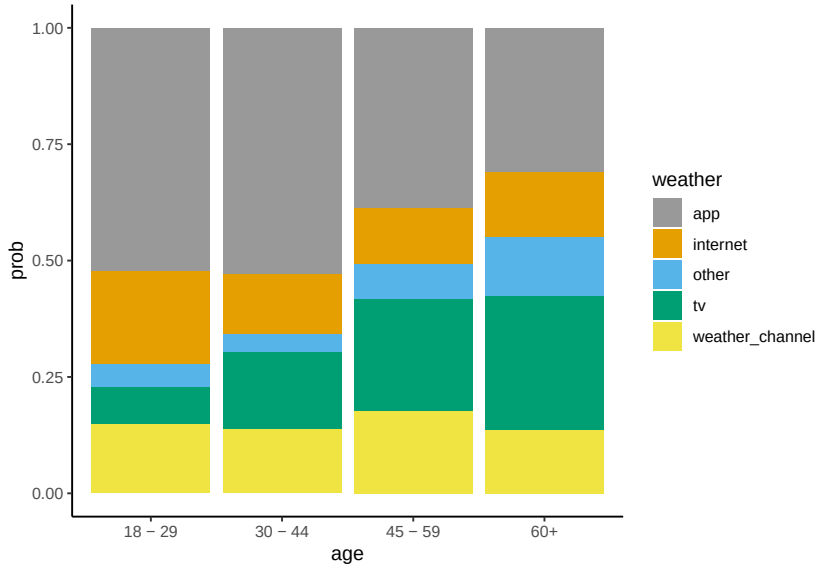

Figure 11: Probability distribution over different sources of weather news as a function of age group.

Bayesian categorical logistic regression

We can perform a Bayesian version of the model in the previous section using `brm` with `family = categorical(link = 'logit')`.

```
M_cat_bayes <- brm(weather ~ age,
                   family = categorical(link = 'logit'),
                   data = weather_df)
```

The summary output is formatted in a long format in comparison to wide format seen above with `multinom`.

```
summary(M_cat_bayes)$fixed
#>                              Estimate Est.Error   l-95% CI    u-95% CI     Rhat Bulk_ESS Tail_ESS
#> muapp_Intercept             2.3681937 0.3456028  1.7258611  3.07083399 1.001543     1189     1776
#> muinternet_Intercept        1.3900579 0.3663141  0.7067108  2.11194174 1.002289     1254     1776
#> mutv_Intercept              0.4512383 0.4309300 -0.3917491  1.27590583 1.002892     1354     2070
#> muweatherchannel_Intercept  1.0932604 0.3841146  0.3496441  1.84002398 1.001782     1270     1752
```

```
#> muapp_age30M44              0.2616396 0.5025598 -0.6793213  1.27311116 1.002277     1365     1775
#> muapp_age45M59             -0.7249908 0.4203623 -1.5574123  0.09504825 1.002130     1411     2088
#> muapp_age60P               -1.4856187 0.4025590 -2.2626366 -0.71451503 1.002007     1398     2053
#> muinternet_age30M44        -0.1979753 0.5396543 -1.2392054  0.87074519 1.001707     1514     1848
#> muinternet_age45M59        -0.9472343 0.4579603 -1.8520972 -0.05810783 1.002109     1548     2495
#> muinternet_age60P          -1.3162809 0.4376876 -2.1836292 -0.47309898 1.001769     1539     2293
#> mutv_age30M44               1.0099413 0.5791534 -0.1202443  2.16831825 1.001522     1398     2075
#> mutv_age45M59               0.7091732 0.4936254 -0.2720899  1.67957014 1.002489     1545     2207
#> mutv_age60P                 0.3526576 0.4755895 -0.5502657  1.30851344 1.003199     1483     2250
#> muweatherchannel_age30M44   0.1734805 0.5555558 -0.9085286  1.27491266 1.000852     1411     1817
#> muweatherchannel_age45M59  -0.2450089 0.4601888 -1.1567660  0.64638359 1.002080     1476     2001
#> muweatherchannel_age60P    -1.0443862 0.4569367 -1.9499843 -0.17354991 1.001375     1584     2110
```

We can rearrange the posterior mean estimates as follows.

```
fixef(M_cat_bayes) %>%
  as_tibble(rownames = 'var') %>%
  dplyr::select(var, Estimate) %>%
  separate(var, into = c('var', 'age')) %>%
  pivot_wider(var, names_from = age, values_from = Estimate)
#> # A tibble: 4 x 5
#>   var              Intercept age30M44 age45M59 age60P
#>   <chr>                <dbl>    <dbl>    <dbl>  <dbl>
#> 1 muapp                2.37     0.262   -0.725 -1.49
#> 2 muinternet           1.39    -0.198   -0.947 -1.32
#> 3 mutv                 0.451    1.01     0.709  0.353
#> 4 muweatherchannel     1.09     0.173   -0.245 -1.04
```

Just as above, with the Bayesian model, we can perform predictions using `predict` or `add_predictions`. Note that here we do not need to use `type = 'probs'`.

```
tibble(age = c('18 - 29', '30 - 44', '45 - 59', '60+')) %>%
  add_predictions(M_cat_bayes)
#> # A tibble: 4 x 2
#>   age     pred[,"P(Y = other)"] [,"P(Y = app)"] [,"P(Y = internet)"] [,"P(Y = tv)"] [,"P(Y = weather_channel)"]
#>   <chr>                   <dbl>           <dbl>                <dbl>          <dbl>                        <dbl>
#> 1 18 - 29                0.0455           0.535                0.193          0.082                        0.144
#> 2 30 - 44                0.0365           0.523                0.134          0.164                        0.144
#> 3 45 - 59                0.0707           0.394                0.124          0.235                        0.176
#> 4 60+                    0.135            0.309                0.138          0.283                        0.135
```

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.

Venables, W. N., and B. D. Ripley. 2002. *Modern Applied Statistics with S*. Fourth. New York: Springer.