Applied Research Methods

Workshop 7 - Logistic regression

In Workshop 6, we worked through an example of linear regression. Linear regression is a modelling approach where the aim is to describe mathematically the relationships between variables - and, in particular, to derive an equation that allows us to predict one variable (the dependent variable) using other variables (independent variables).

Linear regression is applicable when the dependent variable is numerical, but there are many contexts where this is not the case. One particularly common situation is where the dependent variable is binary - i.e. it only takes two possible values (which we generically think of as 0 and 1). In these circumstances, linear regression will not be valid, and so we need another technique - logistic regression. We introduced the theory and principles behind this in the Lecture on logistic regression.

In this workshop, we will put logistic regression into practice by working with survey data. In particular, we will attempt to predict individuals' attitudes towards crime on the basis of their other characteristics.

Modelling worry about crime

As we have seen in previous workshops, the Crime Survey for England & Wales (CSEW) is a rich source of data about attitudes towards, and experience of, crime. Its large scale and high-quality design mean that it is a representative measure of these issues for the population of England & Wales, and it is widely used in both research and policy contexts.

The CSEW contains a number of questions relating to the fear of crime, and we have studied several of these in previous workshops. Most of these questions ask respondents how worried they are about a certain type of crime (e.g. rape, car theft), with the responses recorded on a Likert-type scale ranging from 'not at all worried' to 'very worried'. In previous workshops, we examined the relationships between some of these variables and other socio-demographic factors. These suggest that levels of worry are dependent on other characteristics - or, put differently, that other characteristics have *predictive value* with respect to worry - and we are going to investigate this further here.

In particular, we are going to examine the extent to which individuals are worried about burglary (we could have chosen any offence type but the choice of offence is unimportant for the purposes of illustration), and try to understand how the level of worry is related to a number of other characteristics. In particular, we are going to examine how well an individual's worry about burglary can be predicted by the following characteristics (our **independent variables**):

- Sex
- Ethnicity
- Whether they live in an urban or rural area

To begin, read in the file 'csew1314teachingopen.csv' which we used in a previous workshop:

```
> csew<-read.csv("CSEW1314teachingopen.csv")
```

The variables in the dataset that are relevant for our analysis are summarised in the following table. As well as the variable definitions, the table shows the values and their corresponding labels - this information can be found in the codebook.

Variable	Content	Values
Sex	Sex of respondent (2 categories)	Male Female
ethgrp2a	Ethnic group (5 categories)	White Mixed Asian or Asian British Black or Black British ChineseorOther
Rural2	Type of area 2004 (2 categories)	Urban Rural
wburgl	How worried about having your home broken into (7 categories)	Very Worried Fairly Worried Not very worried Not at all worried Not applicable Refusal Don't know

Data preparation

Although we have all the information that we need in order to investigate these relationships, the variables are not currently in such a form that we can apply logistic regression. Before building a model, therefore, we first need to perform some pre–processing of the data.

Recoding the dependent variable

Logistic regression is applicable in situations in which the dependent variable is binary (or dichotomous) - that is, it has two possible values. In the analysis of crime and security, this often represents some event either happening or not doing so, such as a victimisation. While it is possible to build models to predict a categorical outcome with 3 or more values, their wide applicability and (relatively) simple interpretation means that binary logistic regression models are much more commonly applied.

At the moment, our dependent variable 'wburgl' is not binary - as seen in the table above, it has 7 possible values (actually only 4 of these are meaningful; the remaining 3 are various forms of missing data). In order to apply logistic regression, therefore, we will first convert it to a binary variable. We will do this by combining categories. In this case, we will collapse the first two categories ("Very worried" and "Fairly worried") and the last two ("Not very worried" and "Not at all worried"), coding them as one and zero, respectively. A value of one will thus indicate that the respondent expressed some degree or worry, while

a value of zero indicates that they did not. To do this we will create a new variable, “bWorry”. We will start by telling R to create the new variable, generating as many observations as there are in the csew dataframe. We will also assign all observations the value of zero. By doing the latter, we can then just recode the values that need to contain the value “1”.

To do this, let’s start by counting the number of observations in the dataframe csew:

```
> l<-length(csew$rowlabel)
```

This stores the length of the dataframe in the variable “l” (we could have called this variable anything). Specifically, the “length” command counts how many observations there are of the variable “rowlabel”. We could have asked R to count the length of any of the variables in the dataframe – I chose this one as it is the first variable in the dataframe. The “rep” command below then tells R to create a new variable called “bWorry” of length “l” (which is 8843) and to assign each observation the value of zero.

```
> bWorry<-rep(0,l)
```

To keep things neat, let’s now add this variable to the csew dataframe by typing:

```
> csew$bWorry<-bWorry
```

We can now record the variable for those observations that respondents expressed some degree of worry. To do this, type:

```
> csew$bWorry[(csew$wburgl == "Fairly worried")|(csew$wburgl == "Very worried")] <- 1
```

The command asks R to assign a value of 1 to an observation when the condition in the square brackets is met. We have two conditions in the square brackets. The first is whether the value for the variable “csew$wburgl” is equal to “Fairly worried”, the second is whether the value for the same variable is “Very worried”. The “|” symbol is equivalent to the logical “OR” command. So, the command tells R to assign the value of one to an observation for the variable bWorry if the value for the variable wburgl is equal either to “Fairly worried” or “Very worried”.

To check this has worked, we can inspect several values for each variable. One approach is to list the first (say) 100 observations for each variable to check they are coded as expected. To this, type:

```
> csew$wburgl[1:100]
```

```
> csew$bWorry[1:100]
```

I won't list the output here as it is quite long, but you should see that the fourth and fifth observations are labelled “Fairly worried” for the original variable and coded as 1 for the new variable. Alternatively, we could ask R to list specific cases. For example, type:

```
> csew$wburgl[73]
```

```
> csew$bWorry[73]
```

In this case, you should get the following output, indicating that for observation 73 the label for the original variable is “Very worried” and the value for the new variable is 1 (which is what it should be):

```
> csew$wburgl[73]
[1] "Very worried"
> csew$bWorry[73]
[1] 1
```

The only thing we need to be careful about in taking this approach is that many of the observations of the "wburgl" variable were missing, but we have coded all of our data for our new variable as zeros or ones. The latter implies that we have no missing data, which is incorrect. This is not a problem but we need to ensure we do not analyse all of the data assuming that there is no missing data. To do this, let's copy the data for which we have complete information for the dependent variable into a new dataframe "csewL". To do this, use the approach we employed in a previous workshop:

```
> csewL<-csew[which(csew$wburgl!=""),]
```

Recall that the `!=` means not equal to. Observations that contain the value "" are missing data so we are just telling R to only use rows of data for which there are no missing data. The new dataframe should have 2194 observations. Check this using the "length" command.

Dummy coding an independent variable

In logistic regression (or indeed any other type of regression), the independent variables must be numerical - clearly an equation of the form $\beta_0 + \beta_1 x$ only makes sense if x is a number... This presents a problem if we wish to use a categorical variable, such as 'Ethnicity', as a predictor - we cannot do calculations with labels like 'White', 'Black', etc. So how can we incorporate such a variable in a model?

Thankfully, there is a neat trick, called **dummy coding**, which allows us to represent a categorical variable numerically. This involves creating a series of new variables: for each category of the original variable, we create a dichotomous variable (i.e. taking a value of 0 or 1) indicating whether the value is equal to that category (1 for 'yes', 0 for 'no'). For example, if we had a variable 'Weapon' with possible values 'Gun', 'Knife' and 'Other', we would create three indicator variables: one for 'Gun (yes/no)', one for 'Knife (yes/no)' and one for 'Other (yes/no)'.

Weapon	Gun	Knife	Other
Gun	1	0	0
Knife	0	1	0
Other	0	0	1

For any case, exactly one of the indicator variables will be equal to 1, while the rest will all equal 0. These three new variables carry exactly the same information as the original categorical variable - if we know the values of the three variables, we know the value of 'Weapon', and vice versa. Strictly, in fact, one of the indicator variables is redundant - we could remove the 'Other' variable, for example, and would still be able to identify 'Other' cases because they would be the ones for which the 'Gun' and 'Knife' values are equal to 0.

The reason for performing dummy coding is because the resulting indicator variables are numerical, and so can be used as inputs in a regression equation. As we will see later, we can interpret the resulting findings in terms of the original categorical variable. To be clear, this procedure is not specific to logistic regression - the same idea applies whenever we want to use a categorical variable in a linear regression too.

In our case, we need to dummy code the 'ethgrp2a' variable, which is a categorical variable with 5 possible values. To do this, we follow the same approach as above, creating new variables, one for each type of ethnicity:

```
m<-length(csewL$wburgl)
white<-rep(0,m)
mixed<-rep(0,m)
asian<-rep(0,m)
black<-rep(0,m)
chineseorOther<-rep(0,m)
csewL$white<-white
csewL$mixed<-mixed
csewL$asian<-asian
csewL$black<-black
csewL$chineseorOther<-chineseorOther
```

This is a little tedious but it works. Moreover, if you store these commands (and others) in a script file, you can reuse them, or variants of them. Now we have to code the values. First, let's check how the ethnicity variable was originally coded so that we know what labels we need to use to recode the data. To do this type the command below, which should produce the output beneath it:

```
> table(csewL$ethgrp2a)
```

```
> table(csewL$ethgrp2a)

                    Asian or Asian British Black or Black British       Chinese or Other                  Mixed                  White
                 2                    103                     65                     25                     20                   1979
```

```
csewL$white[csewL$ethgrp2a == "White"] <- 1
csewL$mixed[csewL$ethgrp2a == "Mixed"] <- 1
csewL$asian[csewL$ethgrp2a == "Asian or Asian British"] <- 1
csewL$black[csewL$ethgrp2a == "Black or Black British"] <- 1
csewL$chineseorOther[csewL$ethgrp2a == "Chinese or Other"] <- 1
```

To check that the variables have been coded correctly, we can manually inspect the observations (see above). We could also count how many observations were coded to each ethnicity. To do this for those coded as "white" type:

```
> sum(csewL$white)
```

This should return the value 1979. We also need to recode the "sex" and "rural" variables so that they are numerical values. We do the same as before – create new variables and code them accordingly:

```
sexB<-rep(0,m)
csewL$sexB<-sexB
csewL$sexB[csewL$sex == "Male"] <- 1

ruralB<-rep(0,m)
csewL$ruralB<-ruralB
csewL$ruralB[csewL$rural2 == "Urban"] <- 1
```

Some of you may have noticed that when we produced the table for ethnicity, there were two observations without a value. As these are missing data, let's produce a final dataset ("csewL2") that excludes these observations:

```
csewL2<-csew[which(csewL$ethgrp2a!=""),]
```

Running logistic regression

We now have our data in a suitable format to run a model - we have a binary dependent variable and a series of numerical independent variables. Let's run the full model. To do this type:

```
model<-
glm(csewL2$bWorry~csewL2$sexB+csewL2$ruralB+csewL2$white+csewL2$mixed+cs
ewL2$asian+csewL2$black, binomial)

summary(model)
```

The text for the first command that begins "model<-" should all be on one line. The "glm" command is the general linear model command, which is equivalent to the "lm" command we used for the linear regression in the previous workshop. As we want to compute a logistic regression rather than OLS, we specify that we are modelling a binomial distribution. Otherwise, the command is structured the same as for the linear regression we have used previously. This should generate the following output:

```
Call:
glm(formula = csewL2$bWorry ~ csewL2$sexB + csewL2$ruralB + csewL2$white +
    csewL2$mixed + csewL2$asian + csewL2$black, family = binomial)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.5505  -1.0233  -0.8864   1.3367   1.6562

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)   -0.383512   0.418608  -0.916 0.359583
csewL2$sexB   -0.347478   0.090820  -3.826 0.000130 ***
csewL2$ruralB  0.364373   0.109082   3.340 0.000837 ***
csewL2$white  -0.347825   0.407894  -0.853 0.393807
csewL2$mixed  -0.009634   0.606708  -0.016 0.987330
csewL2$asian   0.863747   0.455594   1.896 0.057978 .
csewL2$black   0.386527   0.476221   0.812 0.416989
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2897.2  on 2191  degrees of freedom
Residual deviance: 2822.6  on 2185  degrees of freedom
AIC: 2836.6

Number of Fisher Scoring iterations: 4
```

Model coefficients

The output relating to each independent variable is shown in the middle part of the output. Before discussing the coefficients themselves, though, it is worth noting one particular feature of the model. That is, we omitted the variable for 'Chinese Or Other' from the model. The reason for this relates to something we mentioned above when we introduced dummy coding. When we dummy code a categorical variable, one of the indicator variables is actually redundant because its values can be worked out from the other indicator variables. In this case, we can tell the value of 'Ethnicity_ChineseOrOther' by looking at the other 4 indicators - if they are all 0, 'Ethnicity_ChineseOrOther' must be equal to 1, whereas if any of the others are equal to 1, 'Ethnicity_ChineseOrOther' must be 0.

From a modelling perspective, this is a problem, because it is an example of collinearity - there is an association (in this case a perfect one) between the independent variables. To deal with this, we omit one of the problematic variables from the model - in this case, we (arbitrarily) chose to remove the 'Ethnicity_ChineseOrOther' variable, but any of the others could equally well have been removed. If this seems confusing, consider that for the gender and urban/rural variables, we also only had one variable. For example, we did not have a variable that was coded “1” for males and a variable coded “1” for females. Omitting one variable enables us to model the effect of the k-1 levels of the variable (where k is the number of levels, two for gender with our data), relative to a baseline category.

Moving on to look at the remainder of the output table, we can see a number of values for each independent variable: as well as the coefficient estimate, we see

- The standard error, which is a measure of the uncertainty associated with the coefficient estimate
- The z score (also known as the Wald statistic), which is the number of standard errors that the coefficient is away from 0
- The p-value to determine the significance of the coefficient estimate

The significance of the variables can be interpreted in just the same way as it was for linear regression. Inspecting the results, we can see that we have two variables that are significant at the $p<0.05$ level: 'sexB' and 'ruralB'. The 'asian' variable is also close to being significant, but is not quite on the right side of the threshold.

By examining the coefficients themselves, we can understand the direction of the predicted effects. The fact that the 'sex' coefficient is negative, for example, means that it is negatively associated with the outcome - i.e. individuals with higher values of 'sex' are less likely to be worried. To understand what this means in real-world terms, we need to remind ourselves what the values of 'sex' represent: as per the table near the start of the workshop, 1 represents 'Male' and 0 represents 'Female'. A one-unit increase in 'sex' therefore corresponds to a change from 'Female' to ‘Male’, and we can interpret the coefficient in these terms: female respondents are more likely to be worried than males. In particular, the log odds of worry are -0.348 lower for males than for females.

The coefficient for 'ruralB', on the other hand, is positive. Because 1 represents 'Urban' and 0 represents 'Rural', this means that respondents living in urban areas will be more likely to be worried - in particular, their log odds will be 0.365 higher.

Although none of the effects are significant here, it is also important to be aware of how to interpret the coefficients associated with the dummy variables. The key point here is that any effects should be interpreted **relative to the reference category**, which is the category which corresponds to a value of 0 for all of the dummy variables. In our model, 'Chinese or Other' is the reference category, so the coefficients for each of the other dummy variables reflect the change relative to the 'Chinese or Other' group. For example, 'White' respondents are less likely to be worried than 'Chinese or Other' respondents (because the coefficient is negative), whereas 'Asian' respondents are more likely to be worried than 'Chinese or Other' respondents. Note, however, that none of these effects are statistically significant here.

We will explore the interpretation of coefficients further below - for now, we will move on to the other part of the output.

Overall model fit

The values at the bottom of the output relate to the fit of the model as a whole. In the Lecture, we introduced the concept of **deviance**, and explained that two models can be compared by examining the difference between their deviances. In particular, we can assess whether a fitted model offers a significant improvement upon a null (constant only) model by examining the reduction in deviance. We can see that for our model the difference in deviance – relative to the model that contains no independent variables (the null deviance) – is (2897.2-2822.6=) 74.6. As discussed in the lecture, we can determine if this is statistically significant by comparing it to the Chi-Square distribution with the appropriate degrees of freedom. In this case, it is highly significant. To compute the p-value, type:

```
1-pchisq(2897.2-2822.6, 2191-2185)
```

This computes the probability of observing a value at least as large as the difference in deviance values for a given number of degrees of freedom (the difference in degrees of freedom for the null model and that which contains the dependent variables). The small p-value indicates that the model that includes the independent variables provides a better fit to the data than one that includes only a constant.

Odds ratios

As we discussed in the Lecture, the coefficients in a logistic regression model are difficult to interpret in real-world terms. They refer to the additive effect on the log odds of the model outcome, and typically this is not a quantity that has intuitive meaning. In the lecture, we introduced an alternative way of representing this - the odds ratio - which is much easier to interpret. To compute these, we simply exponentiate the log (odds) values.

Odds ratios provide a means to understand the relationship between independent variables and the outcome. As with linear regression, they correspond to the effect of a one-unit increase in the independent variable; however, unlike linear regression, the effect is multiplicative (it is multiplicative because we are working with exponentiated values) and it relates to the odds. Put simply, **a one-unit increase in the**

independent variable results in the odds of the outcome being multiplied by the corresponding odds ratio.

Expressed in this way, we can outline a few general principles when interpreting odds ratios:

- An odds ratio of 1 corresponds to 'no effect', because multiplying the odds by 1 just leaves them unchanged
- An odds ratio greater than 1 indicates a positive relationship, because it means that a one-unit increase in the independent variable will cause the odds to increase (if something is multiplied by a number greater than 1, it increases)
- An odds ratio less than 1 indicates a negative relationship, because it means that a one-unit increase in the independent variable will cause the odds to decrease

If we compute the odds ratio associated with 'sex' in our data by calculating $e^{-0.347}$, by typing:

```
exp(-0.347)
```

we see that it is 0.71 - this means that sex is negatively associated with the outcome. As before, in order to make sense of this, we need to know how 'sex' is coded. Since 1='Male' and 0='Female' - meaning that a one-unit increase in the independent variable corresponds to a change from 'Female' -> 'Male' - we know that this means that – relative to females – males are less likely to be worried about crime.

We can also express this change as a percentage. The odds ratio implies that the odds of being worried for males is 0.71 times the odds of being worried for males, which represents a 29% decrease - if you multiply a number by 0.71, you decrease it by 29%... In general, odds ratios can be expressed in percentage terms by applying the following formula:

$$\% \text{ increase} = 100 \times (\text{Odds Ratio} - 1)$$

We can do the same thing for the 'Ethnicity_White' variable. In this case, the odds ratio is less than 1 - it is 0.706- which tells us that the association is negative. Since, for this variable, a value of 1 represents 'White' respondents, and the reference category is 'Chinese or Other', this means that 'White' individuals are less likely to be worried about crime than individuals with 'Chinese or Other' ethnicity. Applying the formula above indicates that this represents a percentage increase of -29.4% - in other words, a decrease of 29.4%. So the odds of being worried are 29.4% lower for 'White' residents than for 'Chinese or Other' residents.

Classification

By this point, we have quite a good understanding of the relationships between our independent variables and our outcome: we know which variables have a significant influence, and we know how they influence the outcome. Remember, though, that ultimately the objective of logistic regression is to predict the dependent variable; that is, to predict whether an individual will be worried about burglary on the basis of their characteristics, or not. We can also evaluate the model's performance explicitly in terms of how accurately it predicts the outcomes.

How can we use the model to predict whether an individual will be worried about burglary? Well, remember that our model essentially provides us with a formula to predict the odds of being worried

about crime, given values for the independent variables. Since odds are just another way of expressing probabilities, we effectively have a formula for predicting the probability that an individual will be worried about crime.

We can use these probabilities to make predictions by adopting a classification rule. For example, we could say that if the predicted probability is greater than 50% then we will predict that the individual will be worried about crime - on the other hand, if the predicted probability is less than 50%, we will predict that they will not be worried. We are essentially saying that we will predict whichever outcome the model suggests is more likely. As an aside, a probability of 50% is equivalent to an odds of 1, so we could also express the classification rule in terms of having odds either more or less than 1.

As far as I am aware, there is not a simple way of doing this with the R base package (you can find packages that will do it for you), but it is not hard to produce this type of table. To do this, we first extract the predicted values from the model. However, we want the probabilities, and by default R will provide the log of the odds. To transform these, we exponentiate the log odds to get the odds and then convert these to probabilities using the formula discussed in the lecture:

$$P(A) = \frac{Odds(A)}{1 + Odds(A)}$$

In R:

```
pred<-(exp(predict(model))/(1+exp(predict(model))))
```

Where the text highlighted yellow is the odds for the predicted values. Now we have these, we have to convert them to ones (for predicted probabilities above 0.5) and zeros (for predicted values equal to or below 0.5). First, we create a new variable to store the zeros and ones called “pred1”, we then assign values to this variable depending upon the predicted values (which we computed for the variable “pred”).

```
t<-length(csewL2$bWorry)

pred1<-rep(0,t)

pred1[pred > 0.5] <- 1
```

Now we have these values, we can compare them to the actual values by cross tabulating them as follows:

```
ftable(pred1, csewL2$bWorry)
```

This should produce the following output:

```
          0    1
pred1
0      1309  715
1        64  104
>
```

This tabulates all 2192 values according to whether a zero was predicted and observed (top left cell), whether a zero was predicted but a value of one observed (top right cell), whether a value of one was predicted and observed (bottom right cell), and whether a value of one was predicted but a zero observed

(bottom left cell). That is, the rows relate to the levels of the predicted values (zero or one) and the columns to those observed (zero or one). If we changed the order in which the variables are listed in the table command, the rows and columns would be reversed.

Another way of thinking about the tabulation is that the numbers correspond to both correct and incorrect predictions. For example, the bottom right cell (containing 104) corresponds to cases that were predicted as being worried, and were indeed worried - these are correct predictions; specifically **true positives**. The bottom left cell, on the other hand, refers to cases that were predicted as being worried but actually were not - these are **false positives**. We can also see that the model predicted that 715 individuals would not be worried when they actually were (**false negatives**), but correctly predicted non-worry in 1,309 cases (**true negatives**). In a perfect model, of course, we would have no false positives and no false negatives.

These numbers can be used to express model performance as classic metrics used for classification systems:

- **Sensitivity** - the percentage of positive outcomes correctly predicted
- **Specificity** - the percentage of negative outcomes correctly predicted
- **Positive predictive value (PPV)** - the percentage of cases predicted to be positive that actually are positive
- **Negative predictive value (NPV)** - the percentage of cases predicted to be negative that actually are negative

For example, the sensitivity of our model indicates that only ($\frac{104}{104+715}$ =) 12.7% of worried people would be correctly identified by our model. The PPV, on the other hand, indicates that if our model predicts that someone will be worried, there is only a ($\frac{104}{104+64}$ =)) 61.9% chance that they actually are.

Overall, the model correctly predicts 1413 of 2192, or 64.5% of outcomes. Is this any good? Well, if we had predicted that the most likely outcome was the most commonly occurring value (i.e. the mode), we would have been correct for 1373 observations (ie the most common value observed was zero and this occurred 1373 times). This would generate an accuracy rate of 62.6%. This is the level of accuracy we would achieve for a model excluding any independent variables. Given that the two levels of accuracy are similar, our model is not very good, even though it is statistically significant.

Reporting

To sum up, one possible way of writing about these results would be to say:

> *A logistic regression analysis showed that, in combination, the independent variables significantly impacted on people's likelihood to worry (or not) about burglary ($\chi 2(6)= 74.6$, $p<0.001$) and the model correctly predicted 64.5% of the responses. A number of individual variables were significant predictors of levels of worry about burglary. There was a significant relationship between sex and levels of worry about burglary (exp(coeff)= 0.71, z=-3.83; p<0.001), with males being less likely to worry about burglary than females. In addition, there was a significant relationship with the type of area (exp(coeff)= 1.44, z=-3.34; p=0.001), with residents of urban areas being more likely to worry about burglary.*