CHAPTER 13

Multinomial (polytomous) logistic regression models of three and more near synonyms

What you will learn from this chapter:

This chapter continues the discussion of logistic regression models, which can be used to predict the speaker's choice between different near synonyms or variants. This time you will learn to model situations when the number of possible outcomes is greater than two. Such models are called multinomial, or polytomous. The method will be illustrated with a case study of three near synonyms: *let*, *allow* and *permit*.

13.1 What is multinomial regression?

A quick check in any dictionary of synonyms is sufficient to see that a pair of semantically similar words is rather an exception than a rule. Mostly, synonyms come in larger groups. For more than two possible outcomes you will need a special type of logistic regression, which is called multinomial, or polytomous. Multinomial models can be classified into two types. One type involves fitting two or more binary models, which compare each outcome against some reference level. Consider three near synonyms *let*, *allow* and *permit*. It is possible to model the choice between *permit* and *let*, and that between *allow* and *let*, with *let* as the base for comparison. The other approach models the odds of each outcome against all other outcomes ('one vs. rest'). For example, it can tell us which factors increase the chances of *let* against both *allow* and *permit*, the chances of *allow* against both *let* and *permit*, and the chances of *permit* against both *allow* and *let*. In this chapter, both approaches will be discussed and their results will be compared.

13.2 Multinomial models of English permissive constructions

13.2.1 Data and hypotheses

You will need several add-on packages, which contain the data and functions that are required for this case study. These packaged should be installed and then loaded.

```
> install.packages(c("mlogit", "polytomous"))
> library(Rling); library(mlogit); library(polytomous)
```

This case study will investigate the differences between three English permissive constructions *let*, *allow* and *permit* + (*to*) Infinitive. The data represent a random sample of the three constructions from COCA (Davies 2008 –). Since *let* is not used in passive contexts (**She was let leave*), only active forms will be considered. The sample is drawn from two registers of COCA: spontaneous conversations (transcripts from diverse TV and radio programmes) and popular magazines.

The variables reflect the following information:

- *Verb* is the response variable with three values: 'allow', 'let' and 'permit'.
- *Permitter* is the semantic class of the matrix clause subject: 'Anim' for animate, 'Inanim' for inanimate and 'Undef' for undefined.
- *Imper* reflects the mood of *let*, *allow* or *permit*: 'Yes' (imperative) or 'No' (all other forms).
- *Reg* is the register: 'Spok' for conversations and 'Mag' for magazine articles.
- *Year* is the year when the source text was published or broadcast, from 1990 to 2012.

The dataset with over 500 examples is available as the data frame `let`.

```
> data(let)
> str(let)
'data.frame': 518 obs. of 5 variables:
$ Year: int 2003 2005 1990 2007 1997 1996 1995 2007 2005 1992 …
$ Reg: Factor w/ 2 levels "MAG","SPOK": 1 2 2 2 1 1 1 2 1 2 …
$ Verb: Factor w/ 3 levels "allow","let",..: 1 1 2 1 3 1 1 2 3 1 …
$ Permitter: Factor w/ 3 levels "Anim","Inanim",..: 2 2 1 2 1 1 2
1 1 2 …
$ Imper: Factor w/ 2 levels "No","Yes": 1 1 1 1 2 1 1 2 1 1 …
```

The constructions and the corresponding permissive verbs have similar conceptual contents that can be described as non-occurrence or cessation of impingement. However, they are not identical. As noted by Wierzbicka (2006), *let* is used in many situations of cooperative interaction, as in *let me help you*, *let me know* or *let me see,* which involve the imperative use. This is why one can expect that *let* will have higher chances to be used in the imperative than the other verbs. Second, *permit* and *allow* can be used in the meaning 'to provide an opportunity, make possible', as in (1) and (2), whereas *let* would not be appropriate in such contexts. Therefore, we can expect *permit* and *allow* to be used with non-human Permitters more frequently than *let*.

(1) *GEEK TRIVIA. Tesla Originally Believed His Radio **Allowed** Him To Communicate With Whom? The Dead. Astronauts. Sailors. Martians. Answer: Martians.* (http://www.howtogeek.com, accessed 3.03.2014)

(2) *Not seeing people **permits** us to imagine them with every perfection.* (Victor Hugo)

In addition, *let* sounds the least formal of these three alternatives, whereas *permit* is the most formal verb, which can be used to denote official authorization. This is why *permit* is expected to have the highest odds of occurrence in the texts from magazines, and *let* to have the lowest ones.

13.2.2 Contrasting *allow* and *permit* with *let*

This subsection demonstrates how a multinomial model which contrasts each outcome with the reference level can be fitted and interpreted. For fitting the model, we will use the package `mlogit`. This function requires a special 'long' data format. In this format, every observation is repeated several times for every possible outcome. To transform a data frame in the original wide format, where each outcome is represented only once, into the long format, one can use the function `mlogit.data()` from the same package. It requires the following arguments: the original data frame, its current shape (`shape = "wide"`) and the outcome variable (`choice = "Verb"`). The resulting object contains three times more rows than the original data frame because every observation is repeated three times, for each of the possible three outcomes:

```
> let1 <- mlogit.data(let, shape = "wide", choice = "Verb")
> head(let1)
          Year  Reg   Verb   Permitter  Imper  chid  alt
1.allow   2003  MAG   TRUE   Inanim     No     1     allow
1.let     2003  MAG   FALSE  Inanim     No     1     let
1.permit  2003  MAG   FALSE  Inanim     No     1     permit
2.allow   2005  SPOK  TRUE   Inanim     No     2     allow
2.let     2005  SPOK  FALSE  Inanim     No     2     let
2.permit  2005  SPOK  FALSE  Inanim     No     2     permit
> nrow(let1)
[1] 1554
```

Let us begin with fitting a model that contains all predictors. The syntax is very similar to other regression functions, with two exceptions. First, one has to add `1|` to create a multinomial model. Second, one can specify the reference level of the response variable. That is the construction that every other synonym will be compared with. By default, the algorithm will select 'allow'. Instead, let us compare *allow* and *permit* with *let*, the second level.

```
> m.let <- mlogit(Verb ~ 1 | Year + Reg + Permitter + Imper, data
= let1, reflevel = 2)
```

One can access the fitted model by using `summary()`:

```
> summary(m.let)
```

```
Call:
mlogit(formula = Verb ~ 1 | Year + Reg + Permitter + Imper, data =
let1, reflevel = 2, method = "nr", print.level = 0)

Frequencies of alternatives:
     let     allow permit
0.36100 0.32239 0.31660

nr method
6 iterations, 0h:0m:0s
g'(-H)^-1g = 1.39E-05
successive function values within tolerance limits.

Coefficients:
                         Estimate    Std.Error  t-value  Pr(>|t|)
allow:(intercept)        0.8184551   47.1686986 0.0174   0.986156
permit:(intercept)       241.1632221 49.3786220 4.8840   1.040e-06 ***
allow:Year               -0.0004914  0.0235595  -0.0209  0.983359
permit:Year              -0.1206881  0.0246883  -4.8885  1.016e-06 ***
allow:RegSPOK            -0.0060516  0.2865426  -0.0211  0.983150
permit:RegSPOK           0.0871446   0.2958475  0.2946   0.768331
allow:PermitterInanim    2.5991669   0.4108197  6.3268   2.503e-10 ***
permit:PermitterInanim   2.6433057   0.4215179  6.2709   3.589e-10 ***
allow:PermitterUndef     1.2653550   0.6067334  2.0855   0.037022 *
permit:PermitterUndef    1.6024125   0.6027837  2.6584   0.007852 **
allow:ImperYes           -3.0887042  0.5453765  -5.6634  1.484e-08 ***
permit:ImperYes          -3.5291867  0.6287956  -5.6126  1.993e-08 ***
--
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Log-Likelihood: -401.94
McFadden R^2: 0.2926
Likelihood ratio test: chisq = 332.5 (p.value = < 2.22e-16)
```

The summary provides a lot of information. The goodness-of-fit statistics are at the bottom. The log-likelihood shows the amount of deviance still left. The smaller the absolute value, the better the fit. McFadden's R^2 is analogous to R^2 in linear regression. Values from 0.2 to 0.4 are considered to indicate a very good fit, which corresponds to 0.7 to 0.9 in linear models (Louviere et al. 2000: 55). With $R^2 = 0.29$ the model fits well. As in binary logistic regression, the likelihood ratio test statistic is used to test the overall significance of the model with the null hypothesis that none of the coefficients is different from zero. A very small *p*-value tells us that the model is significant.

The most unusual part, however, is the table of coefficients. Each predictor appears twice: first, with *allow*, and then with *permit*. Each line compares one of these verbs with

the reference level, *let*. As in binary logistic regression, the coefficients are log odds ratios. A positive coefficient tells us that the given value of the predictor (or every additional unit of measurement, in case of quantitative variables) increases the chances of the given verb in comparison with *let*. A negative coefficient means that the predictor decreases the chances the given verb. For example, the verb *permit* seems to become less frequent with time than *let* (`permit:Year = -0.1206881`). One can get rid of the logarithm and obtain the corresponding simple odds ratio by using `exp()`:

```
> exp(-0.1206881)
[1] 0.8863104
```

Since *Year* is a quantitative variable, the odds of *permit* vs. *let* decrease by the factor of 0.886 with every additional unit of the numerical predictor, that is, with every additional year. Conversely, the odds of *let* against *permit* increase by the factor of $1/0.886 \approx 1.128$ every year. The low *p*-value suggests that this effect is significant.

The table also indicates that inanimate and undefined Permitters significantly increase the odds of *allow* and *permit* against *let*. This means that *let* is a verb that tends to 'prefer' animate semantic subjects, as expected. At the same time, the negative coefficients show that *allow* and especially *permit* are less frequently used in imperative constructions than *let*. Again, this finding is in accordance with the predictions.

Interestingly, the effect of register is not significant at all. It may be that the perceived informality of *let* comes from its association with cooperative interaction and imperative forms, because refitting a model without *Imper* yields significant or marginally significant coefficients of *Reg* (the readers are invited to test this themselves). On the other hand, the spoken conversations found in COCA are in fact quite formal, representing news broadcasts and on-air interviews.

As usual, the 95% confidence intervals of the coefficients can be obtained as follows:

```
> confint(m.let)
                                2.5 %       97.5 %
allow:(intercept)         -91.63049530   93.26740551
permit:(intercept)        144.38290150   337.94354279
allow:Year                -0.04666719    0.04568438
permit:Year               -0.16907626    -0.07229997
allow:RegSPOK             -0.56766482    0.55556163
permit:RegSPOK            -0.49270585    0.66699505
allow:PermitterInanim     1.79397509     3.40435869
permit:PermitterInanim    1.81714593     3.46946552
allow:PermitterUndef      0.07617940     2.45453054
permit:PermitterUndef     0.42097814     2.78384687
allow:ImperYes            -4.15762241    -2.01978601
permit:ImperYes           -4.76160340    -2.29677005
```

It is also possible to obtain fitted values for the actually observed outcomes with the help of `fitted()`:

```
> head(fitted(m.let))
1.let        2.let        3.let        4.let        5.let
0.56121053   0.59260191   0.20854471   0.64121412   0.03173396
6.let
0.26904521
```

The values are the probabilities of the following observed choices:

```
> head(let$Verb)
[1] allow allow let allow permit allow
Levels: allow let permit
```

Relatively small fitted probabilities suggest that the predictive power of the model regarding the individual observations is not so fantastic. Ideally, all fitted probabilities should be close to 1.

It is also possible to obtain fitted values for all three outcomes:

```
> head(fitted(m.let, outcome = FALSE))
      let          allow        permit
[1,]  0.04924250   0.56121053   0.38954697
[2,]  0.05236395   0.59260191   0.35503415
[3,]  0.20854471   0.17673678   0.61471852
[4,]  0.05671517   0.64121412   0.30207071
[5,]  0.93217877   0.03608727   0.03173396
[6,]  0.31648259   0.26904521   0.41447221
```

Note that the probabilities in each row sum up to 1.

Recall that the variable *Reg* did not have any significant effect. Is it redundant? As in the previous regression models, you can use ANOVA to compare two models and see whether adding or deleting one variable has a significant effect. In `mlogit`, one has several options to perform such a comparison. We will use the Wald test here (see the help page of `waldtest()` in the `mlogit` package for two other options).

```
> m.let1 <- mlogit(Verb ~ 1 | Year + Permitter + Imper, data = let1,
reflevel = 2)
```

Next, we compare the less constrained model without *Reg* with the previous one:

```
> waldtest(m.let1, m.let)
Wald test

Model 1: Verb ~ 1 | Year + Permitter + Imper
Model 2: Verb ~ 1 | Year + Reg + Permitter + Imper
```

```
Res.Df Df Chisq Pr(>Chisq)
1    508
2    506 2 0.1658   0.9205
```

The high *p*-value indicates that the variable *Reg* does not improve the fit of the model. This method can be also used for testing potential interactions. There seems to be no significant two-way interactions, as the reader can test him- or herself.[1]

13.2.3 'One vs. rest' approach

The second approach is 'one vs. rest', when each possible outcome is compared with all other possibilities. To fit the model, you will need the package `polytomous`. The possibilities offered by this package are different from `mlogit`. For instance, one can use 'wide' data frames. In addition, the summary returns a variety of goodness-of-fit statistics. To fit a first model with all predictors, you can use the function `polytomous()`:

```
> m.let2 <- polytomous(Verb ~ Permitter + Imper + Year + Reg, data
= let)
```

Let us begin with the goodness-of-fit measures:

```
> summary(m.let2)$statistics
$df.null
[1] 1554

$df.model
[1] 1536

$AIC.model
[1] 845.5114

$BIC.model
[1] 922.011

$loglikelihood.null
[1] -568.1868

$loglikelihood.model
[1] -404.7557

$deviance.null
[1] 1136.374

$deviance.model
```

1. In some cases, the models with interactions did not converge due to a few zeros in the cells of the table of cross-tabulated categorical predictors. Try: `table(let$Permitter, let$Imper, let$Reg)`.

```
[1] 809.5114

$R2.likelihood
[1] 0.2876362

$R2.nagelkerke
[1] 0.5266656

$crosstable

         allow   let   permit
allow    71      42    54
let      8       152   27
permit   34      29    101

$accuracy
[1] 0.6254826

$recall.predicted
allow       let         permit
0.4251497   0.8128342   0.6158537
$precision.predicted
allow       let         permit
0.6283186   0.6816143   0.5549451
[output omitted]
```

The pseudo-R^2 measures are different approximations of the corresponding measure in linear regression. The first R^2 is the McFadden statistic, which was discussed in the previous section. Its value is similar to the one in the previous model. In addition to the measures that you should be already familiar with, there are measures of accuracy, recall and precision. The accuracy measure shows how correctly the model predicts the use of the verbs. To understand what this measure means, consider the numbers in `$crosstable`. The rows display the observed permissive verbs, and the columns show how many times the model would predict each outcome. The numbers in the diagonal show the correct predictions (e.g. the cases when *allow* was both predicted and observed). The statistic `accuracy` then shows the total number of correct predictions divided by the number of observations (71 + 152 + 101)/518 ≈ 0.625. The measure `recall` shows the proportion of instances of each verb, which were predicted by the algorithm. One can see that over 80% of instances of *let* would be predicted as instances of *let* by the model. *Allow*, unfortunately, has a relatively low recall value, only approximately 43%. The measure `precision`, in contrast, shows how many times the predictions of *allow*, *let* or *permit* made by the model were correct. Again, *let* has the highest value over 68%, but now *permit* is predicted the least accurately (about 55%).

Let us now examine the table of regression coefficients. To access it, one can use the code below. Note that by default the function `summary()` will return simple odds and odds ratios. For the sake of compatibility with our previous models, we will represent the estimated coefficients as log odds (ratios).

```
> print(summary(m.let2), parameter = "logodds")
Formula:
Verb ~ Permitter + Imper + Year + Reg
Heuristic:
one.vs.rest

Log-odds:
                  let        allow       permit
(Intercept)       -Inf       -Inf        Inf
ImperYes          3.292      -2.559      -2.954
PermitterInanim   -2.622     0.7804      0.7399
PermitterUndef    -1.411     (0.3325)    0.8294
RegSPOK           (-0.0443)  (-0.05829)  (0.09126)
Year              0.05592    0.07814     -0.1203

Null              deviance:  1136.0 on  1554  degrees  of freedom
Residual (model)  deviance:  809.5  on  1536  degrees  of freedom
R2.likelihood:   0.2876
AIC:             845.5
BIC:             922
```

The table of coefficients shows the log odds ratios of the linguistic and extralinguistic features. Note that if the estimates are in brackets, the coefficients are not statistically significant at the level of 0.05. The actual *p*-values (rounded up to 3 digits after the decimal separator for convenience) can be obtained as follows:

```
> round(m.let2$p.values, 3)
                  let    allow  permit
(Intercept)       0.008  0.000  0.000
ImperYes          0.000  0.000  0.000
PermitterInanim   0.000  0.000  0.001
PermitterUndef    0.012  0.415  0.043
RegSPOK           0.867  0.787  0.684
Year              0.009  0.000  0.000
```

Again, there is evidence that *let* is 'disfavoured' by inanimate Permitters at a significant level (–2.622, $p < 0.001$) in comparison with the two other verbs, whereas both *allow* (0.7804, $p < 0.001$) and *permit* (0.7399, $p = 0.001$) have, in contrast, higher chances to occur when the Permitter is inanimate. These results are similar to the ones discussed in the previous subsection. As for undefined Permitters, they boost the chances of *permit* and decrease the odds of *let*. The estimated coefficient of *allow* is not significant.

The variable *Year* shows how much the log odds of a verb change with every year. The coefficients reveal a significant increase in the use of *allow* with time, followed by *let*, and a decrease in the use of *permit*. With every year, the odds of *permit* seem to decrease by –0.1203 log odds, which makes it 0.886 in simple odds (recall that odds smaller than 1 indicate a decrease).

```
> exp(-0.1203)
[1] 0.8866544
```

This coefficient is nearly identical to the coefficient of `permit:Year` in the `mlogit` model, which compared *permit* with *let*. It seems that *permit* is becoming a rarer verb in this construction in comparison with both *let* and *allow*, since the contrast between *allow* and *let* was not significant in the previous model. This gradual decrease of *permit* with time is somewhat unexpected. Whether this tendency can be traced within a longer period and in other registers is a question for further research.

The effect of register is again not significant, whereas the estimate of *Imper* supports the previous conclusion about *let* as the most 'interactive' verb. However, if all pairwise interactions are added to the model one by one and tested with the help of ANOVA, one can find a significant interaction between *Reg* and *Imper*.[2] The ANOVA results are below:

```
> m.let3 <- polytomous(Verb ~ Permitter + Imper*Reg + Year, data =
let)
> anova(m.let2, m.let3)
Analysis of deviance Table

Model 1: Verb ~ Permitter + Imper + Year + Reg
Model 2: Verb ~ Permitter + Imper *Reg + Year
     Resid. Df Resid. Dev. Df Deviance P(>|Chi|)
Model 1  1536   809.51
Model 2  1533   793.49 3   16.023 0.001122 **
--
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Adding the interaction has changed the goodness-of-fit statistics:

```
> summary(m.let3)$statistics
[output omitted]

$R2.likelihood
[1] 0.3017362

$R2.nagelkerke
[1] 0.544905

$crosstable

         allow   let  permit
allow    80      31   56
let      26      134  27
```

2. Another significant interaction is observed between *Permitter* and *Imper*, but adding it results in data sparseness and 'NA' values due to the fact that some levels of the variables have zero co-occurrence frequency (see the previous footnote).

```
permit   37      23   104

$accuracy
[1] 0.6138996

$recall.predicted
allow      let        permit
0.4790419  0.7165775  0.6341463

$precision.predicted
allow      let         permit
0.5594406  0.7127660   0.5561497
[output omitted]
```

On the one hand, both versions of R^2 have slightly increased. At the same time, the overall accuracy has become slightly poorer. As one can see from the diagonal numbers in the cross-table, the prediction of *allow* and *permit* has in general improved, but the prediction of *let* has become worse (due to the drop in the recall of *let*).

Let us now examine the interaction.

```
> print(summary(m.let3), parameter = "logodds")

Formula:
Verb ~ Permitter + Imper *Reg + Year

Heuristic:
one.vs.rest

Log-odds:
                    allow      let         permit
(Intercept)         -Inf       -Inf        Inf
ImperYes            (-1.058)   1.665       -1.765
ImperYes:RegSPOK    (-16.77)   3.546       (-2.094)
PermitterInanim     0.8013     -2.684      0.7516
PermitterUndef      (0.3611)   -1.486      0.8452
RegSPOK             (0.08194)  (-0.3801)   (0.1598)
Year                0.07736    0.05865     -0.1209

Null deviance:               1136.0  on  1554  degrees of freedom
Residual (model) deviance:   793.5   on  1533  degrees of freedom

R2.likelihood:  0.3017
AIC:            835.5
BIC:            924.7
```

The term `ImperYes` does not show the general effect of *Imper*, but rather the effect of *Imper* when a context comes from a magazine (the reference level). In that case, there is a significant increase of the probability of *let* and a significant decrease of the probability of *permit*, compared to the reference level `ImperNo`. However, in the spoken data the

chances of *let* in imperative constructions are much greater (see the positive interaction term 3.546), and this effect is significant. For non-imperative constructions (the reference level) and the spoken register (the term `RegSPOK`), there seems to be no significant difference between the verbs.

To summarize, the register effect is observed only in imperative contexts. Imperative constructions in the spoken data greatly increase the chances of *let*. What is so special about these constructions? It seems that such structures are very important pragmatically, for example, for argumentation management, which allows to introduce new turns in the topic within one speaker's discourse:

(3) Dr-VENTER: *So Congress is waiting to pass such legislation. But* ***let*** *me give you the other side of this.* (COCA SPOK NPR_Science 2003)

In TV and radio programmes, participants normally have to communicate under time pressure. This is why they have to negotiate, justify or even fight for a chance to say what they have to say, as in the examples below.

(4) SCHIEFFER: *Well,* ***let*** *me just say – and I want to make sure everyone understands this.* (COCA SPOK CBS_FaceNation 1997)

(5) Mr-QUINN: *No,* ***let*** *me –* ***let*** *me finish, Jonathan* (COCA SPOK CBS_FaceNation 1998)

As was mentioned above, such cases can be classified as *let* of cooperative dialogue, following Wierzbicka (2006). They are highly distinctive of *let*, not only in comparison with other verbs in English, but also with the similar verbs in other languages. However, this effect is observed only when *let* is contrasted with **both** *allow* and *permit*. When it is compared to *allow* and *permit* separately, we observe only a borderline significance of the interaction term ($p = 0.053$) for `permit:RegSPOK:ImperYes`, $b = -2.51$.

13.3 Summary

This chapter has introduced multinomial logistic regression analysis. Two approaches to multinomial regression has been discussed: (a) each outcome vs. the reference level and (b) one vs. rest. Although the model interpretation may be quite challenging, the case study demonstrates that the two approaches are complementary and enable one to fine-tune the interpretation of each model.

Writing up the results of multinomial regression

One can use the same approach as in binary logistic regression, but it is recommended to provide two or more tables for each comparison made. It is crucial that you present the estimates and the *p*-values, as well as a goodness-of-fit statistic, such as the McFadden R^2, which is a popular goodness-of-fit statistic for multinomial models.

More on multinomial models

Another linguistic case study where the package `polytomous` is used can be found in Arppe et al. (2013). The R code is provided, as well. The file can be accessed from R by typing in `vignette("shanghainese")`. An example of a `mlogit()` model with interactions is given in Field et al. (2012: Section 8.9). See also Gries (2013: 322–323), who uses another function, `multinom()`.