Regression with the Ordered Stereotype Model (OSM)

Louise McMillan

Contents

Introduction

The `clustord` package is primarily designed for clustering ordinal data, but you can also use it to carry out regression on ordinal response data using the Ordered Stereotype Model (OSM) using the `osm()` function.

The proportional odds model (POM) is the other ordinal model available for clustering in `clustord`, but a regression function is not provided for that, because the `MASS` package already contains such a function, `polr()`. The `osm()` function in `clustord` is designed to be similar to the `polr()` function, with similar arguments and some of the same outputs.

In order to demonstrate the use of the `osm()` function, we will first load the `arthritis` dataset from the `multgee` package. This dataset contains self-assessment scores from rheumatoid arthritis sufferers at an initial baseline assessment and at several follow-up times.

```
library(clustord)
library(multgee)
head(arthritis)
#>   id y sex age trt baseline time
#> 1  1 4   2  54   2        2    1
#> 2  1 5   2  54   2        2    3
#> 3  1 5   2  54   2        2    5
#> 4  2 4   1  41   1        3    1
#> 5  2 4   1  41   1        3    3
#> 6  2 4   1  41   1        3    5
```

The `y` variable is the self-assessment score, which will be our ordinal response. We will convert it to a factor, which is necessary for the analysis. Since the response is already encoded as values 1 to 5, we do not need to change the default factor levels.

```
arthritis$y <- factor(arthritis$y)
```

Ordered Stereotype Model definition

We will define the ordered stereotype model for an observation i with response value Y_i with q levels indexed by $k = 1, \ldots, q$. Y_i depends on covariates $\boldsymbol{X}_i$. The ordered sterotype model takes the form

$$\log\left(\frac{P(Y_i = k \,|\, \boldsymbol{x}_i)}{P(Y_i = 1 \,|\, \boldsymbol{x}_i)}\right) = \mu_k + \phi_k \boldsymbol{\beta}^T \boldsymbol{x}_i$$

where μ_1 is fixed at 0 for identifiability and $0 = \phi_1 \leq \ldots \leq \phi_k \leq \ldots \leq \phi_q = 1$. The restriction on the ϕ_k values is what makes this the **ordered** stereotype model.

The β values are the coefficients of the covariates, as in a linear regression model, except here they change the probability of Y_i taking different levels. If a single coefficient is positive, it increases the probability of Y_i taking higher levels, and if it is negative it increases the probability of Y_i taking lower levels.

It is possible to have a more flexible model where there are separate β coefficients for each level of the response, but that is not available in this package.

Note that this is **not** a cumulative model, nor is it a logistic model, because the $\log$ function on the left hand side of the model equation is not a logit. By contrast, the proportional odds model is a cumulative logit model:

$$\log\left(\frac{P(Y_i \leq k \,|\, \boldsymbol{x}_i)}{P(Y_i > k \,|\, \boldsymbol{x}_i)}\right) = \mu_k - \boldsymbol{\beta}^T \boldsymbol{x}_i$$

Note that the minus sign in this form of the proportional odds model ensures that the coefficients β have the same effect as in the ordered stereotype model, i.e. that positive values of the coefficients increase the probability of Y_i taking higher levels.

The ordered stereotype model is more flexible than the proportional odds model partly because the ϕ_k score parameters change the effect of the coefficients for different levels of the response, so that the model is not proportional at every response level. It is also more flexible because it is not a cumulative model, in that the model is based on the separate relative probability of any of the upper levels compared with the baseline reference level. That is, the ratio in the $\log$ part is a ratio between level k and level 1, without reference to any of the other levels.

Therefore, if you have data that do not fit the proportionality requirements of the proportional odds model, fitting the ordered stereotype model provides more flexibility and only increases the number of independent parameters by $q - 2$ (because ϕ_1 and ϕ_q are already fixed at 0 and 1).

Ordered Stereotype Model fitting

We can fit the ordered stereotype model using a standard regression call, specifying the required formula and the dataset to use. We will test a model in which the response, i.e. the self-assessment score, is dependent on the baseline self-assessment score and the patient's sex and age.

Since the dataset has multiple rows for each patient, containing self-assessment scores at different follow-up time points (i.e. it is a longitudinal dataset), we will select only one of these time points for our analysis, using the `subset` argument of the `osm()` function.

```
fit <- osm(y ~ baseline + sex + age, data=arthritis, subset = (time == 1))
fit
#> Call:
#> osm(formula = y ~ baseline + sex + age, data = arthritis, subset = (time ==
#>     1))
#>
#> Coefficients beta:
#>    baseline         sex         age
#>  2.02851106 -0.62830137 -0.08688262
#>
#> Intercepts mu:
#>         mu1         mu2         mu3         mu4         mu5
#>  0.00000000  2.24005594  3.17757137  2.75084295 -0.01167798
#>
#> Score parameters phi:
#>      phi1      phi2      phi3      phi4      phi5
#> 0.0000000 0.2371290 0.3332959 0.5673667 1.0000000
#>
#> Residual Deviance: 735.9123
#> AIC: 755.9123
#> BIC: 792.9167
#> (3 observations deleted due to missingness)
#> Warning: did not converge as iteration limit reached
```

Notice that the output of this initial fit includes a warning message: `Warning: did not converge as iteration limit reached`. This warning indicates that the `optim()` function inside `osm()`, which is used to fit the model, did not manage to converge to a solution.

We therefore need to refit the model allowing `optim()` to run for more iterations, to ensure convergence. We can do this by passing in the `control` argument for `optim()`, which is a list of control parameters for `optim()`. The only control parameter we will change is the upper limit on the number of iterations, `maxit`. The default is 100, and we will increase it to 5000.

```
fit <- osm(y ~ baseline + sex + age, data=arthritis,
           subset = (time == 1), control=list(maxit=5000))
fit
#> Call:
#> osm(formula = y ~ baseline + sex + age, data = arthritis, control = list(maxit = 5000),
#>     subset = (time == 1))
#>
#> Coefficients beta:
#>     baseline          sex          age
#>  1.913523966  0.832093545 -0.009527915
#>
#> Intercepts mu:
#>       mu1       mu2       mu3       mu4       mu5
#>  0.000000  1.358211  1.057313 -1.918595 -6.584562
#>
#> Score parameters phi:
#>       phi1       phi2       phi3       phi4       phi5
#> 0.00000000 0.03118735 0.23967408 0.64352471 1.00000000
#>
#> Residual Deviance: 719.678
#> AIC: 739.678
#> BIC: 776.6825
```

```
#> (3 observations deleted due to missingness)
```

Now we have got a converged solution, so we can now move on to interpreting the results.

We can show a summary using the standard `summary()` function:

```
summary(fit)
#>
#> Re-fitting to get Hessian
#> Variance-covariance matrix for beta coefficients, mu intercepts and the independent elem
#> Call:
#> osm(formula = y ~ baseline + sex + age, data = arthritis, control = list(maxit = 5000),
#>     subset = (time == 1))
#>
#> Coefficients:
#>              Value Std. Error t value   p value
#> baseline  1.913524     0.3963  4.8282 1.378e-06
#> sex       0.832094     0.5219  1.5942 1.109e-01
#> age      -0.009528     0.0225 -0.4234 6.720e-01
#>
#> Intercepts mu:
#>     Value    Std. Error t value  p value
#> mu2   1.3582   0.3323     4.0879   0.0000
#> mu3   1.0573   0.7852     1.3466   0.1781
#> mu4  -1.9186   1.3093    -1.4654   0.1428
#> mu5  -6.5846   1.9515    -3.3740   0.0007
#>
#> Score parameters phi:
#>      Value    Std. Error t value  p value  t value  p value
#> phi2   0.0312   0.0000   747.9832   0.0000  -2.0856   0.0370
#> phi3   0.2397   0.1000     2.0856   0.0370  -2.6138   0.0090
#> phi4   0.6435   0.1184     2.6138   0.0090  -3.0096   0.0026
#>
#> The first set of t-values and p-values for phi test whether each phi coefficient is sign
#>
#> Residual Deviance: 719.678
#> AIC: 739.678
#> BIC: 776.6825
#> (3 observations deleted due to missingness)
```

Covariate coefficients

The `beta` estimates are the estimated coefficients of the covariates `baseline`, `sex` and `age`. More positive values correspond to a higher probability of getting high response levels (i.e. in this case, you're more likely to get y = 4 or 5 than 1 or 2), and more negative values correspond to a higher probability of getting low response levels (i.e. you're more likely to get y = 1 or 2 than 4 or 5).

We see that the coefficient for baseline is 1.94, which indicates that the baseline score for an individual has a big impact on their follow-up self assessment score. If the baseline increases by one level, then that makes it more likely that the follow-up self-assessment score, y, will also be higher.

We see that the coefficient for sex is 0.84. The reference level, 1, corresponds to female, and 2 corresponds to male. So male patients are more likely to have high self-assessment scores than female patients. But the effect of sex is not as dramatic as the effect of the baseline score.

Finally, we see that the coefficient for age is -0.0090. This is much smaller than the effects of baseline or sex.

It's also slightly negative. So this indicates that every additional year the patient had lived at the start of the trial reduced the chances of them having high follow up scores, but only by a tiny bit.

Overall, the effect sizes suggest that baseline and sex are relevant, but that age is not particularly important.

We can confirm this using the `summary` output. The summary shows approximate p-values for whether each coefficient is significantly different from 0. Any that are not significantly different from 0 indicate that those effects are too small to be detected for this dataset, so if we want to drop those effects from the model, that would not affect the output much (though there is no requirement to drop them).

In this case, age has a p-value above 0.05, which confirms that there is little evidence it is significant, but also the p-value for sex is fairly high, so there is not much evidence of that being significant either. Baseline, however, has strong evidence that it's significant.

`mu` parameters

The `mu` estimates in the output are, roughly speaking, the intercept terms for each level of the response. This is why there is no intercept term amongst the covariate coefficients: it is already incorporated into the `mu` parameters. We do not need to interpret these estimates.

`phi` parameters

The `phi` estimates in the output are the "score" parameters that are unique to the Ordered Stereotype Model. The first and the last values are fixed as 0 and 1.

The values of the remaining coefficients are $\phi_2 = 0.13$, $\phi_3 = 0.18$, $\phi_4 = 0.60$.

Note firstly that ϕ_2 and ϕ_3 have very similar estimated values. The ϕ_k values modify the effects of the covariates so that they have slightly different effects for each level. Since ϕ_2 and ϕ_3 are very similar, covariates have similar effects on both of those levels. Roughly speaking, this means that if the baseline pushes up the chances of getting level 2 scores compared to the reference level 1, the baseline will also push up the chances of getting level 3 scores compared to the reference level 1. There's very little difference in the pattern of the results for response level 2 compared to response level 3.

That tells us that if we want to simplify the scores we could merge levels 2 and 3 without changing the information in the results much.

If we're considering merging levels 2 and 3, we can also check the summary output to see whether the merge would be appropriate. The `summary()` output for ϕ_k parameters shows **two** p-values for each independent value of ϕ_k (ϕ_1 is always 0 and ϕ_q is always 1). Whereas other coefficients are tested to assess whether they are significantly different from 0, ϕ_k values are tested to see whether they are significantly different from the values above or below them. So ϕ_2 has p-values that indicates whether it is significantly different from ϕ_1 or ϕ_3, respectively.

If one of these p-values is above 0.05, that does not mean that those two levels must be merged, it just suggests that there would not be problems in the model if those levels were merged.

But if one of the p-values is very low, and especially if it is below 0.005, then it would be a bad idea to merge that pair of levels because there's strong evidence that they're exhibiting different patterns in the data. In this case the p-value for the comparison between ϕ_2 and ϕ_3 is 0.370 (as seen in the rows for `phi2` and `phi3` in the summary), so there's only fairly weak evidence that they're different. We probably wouldn't merge them in this situation, but let's try it now to illustrate what effect that would have on the results.

Merging response levels

We start by constructing a new version of the response with both of those levels converted to level "2.5" and defining that as a new factor:

```r
arthritis$y_merged <- as.numeric(as.character(arthritis$y))
arthritis$y_merged[arthritis$y_merged %in% c(2,3)] <- 2.5
arthritis$y_merged <- factor(arthritis$y_merged)

fit_merged <- osm(y_merged ~ baseline + sex + age, data=arthritis,
                  subset = (time == 1), control = list(maxit = 5000))
summary(fit_merged)
#>
#> Re-fitting to get Hessian
#> Variance-covariance matrix for beta coefficients, mu intercepts and the independent elem
#> Call:
#> osm(formula = y_merged ~ baseline + sex + age, data = arthritis,
#>     control = list(maxit = 5000), subset = (time == 1))
#>
#> Coefficients:
#>             Value Std. Error t value   p value
#> baseline  1.8494    0.50359   3.672 0.0002403
#> sex       0.8025    0.55857   1.437 0.1508209
#> age      -0.0247    0.02362  -1.046 0.2956452
#>
#> Intercepts mu:
#>       Value   Std. Error t value p value
#> mu2.5  2.0829  0.9111     2.2861  0.0223
#> mu4   -1.1192  1.3701    -0.8169  0.4140
#> mu5   -5.6486  2.0110    -2.8089  0.0050
#>
#> Score parameters phi:
#>        Value   Std. Error t value p value t value p value
#> phi2.5  0.1477  0.1617     0.9135  0.3610 -3.1599  0.0016
#> phi4    0.6101  0.1496     3.1599  0.0016 -2.6069  0.0091
#>
#> The first set of t-values and p-values for phi test whether each phi coefficient is sign
#>
#> Residual Deviance: 510.7329
#> AIC: 526.7329
#> BIC: 556.3365
#> (3 observations deleted due to missingness)
```

As you can see, the coefficient estimates for the covariates have changed a little, but the general pattern is the same, because levels 2 and 3 were not giving us much different information about the covariates anyway.

If we wanted to simplify the data even further we could consider merging response levels 1 and 2.5, which are still fairly close together, but that's probably unnecessary. ϕ_4 is very different to both $\phi_{2.5}$ and ϕ_5, so that suggests those three levels are all useful for understanding the effects of the covariates, so we should not merge them.

Generally speaking, we would consider any difference between consecutive ϕ_k values that's smaller than 0.1 to be a small difference, any difference between 0.1 and 0.2 to be a fairly small difference and any difference above 0.2 to be a large difference. Levels with ϕ_k values more than 0.2 apart should not be merged.

The p-values in the summary should be taken as guidance, but not definitive, since if we have a very large dataset even small changes can appear significant. For a dataset bigger than 5000 observations it would not be surprising to find every level of ϕ_k is significantly different than every other, because that quantity of data would make it easier to spot subtle differences, even if those subtle differences are not of practical significance.

In the small arthritis dataset, f we merged levels 4 and 5 we'd expect the results to change a lot:

```
arthritis$y_merged2 <- as.numeric(as.character(arthritis$y_merged))
arthritis$y_merged2[arthritis$y_merged2 %in% c(4,5)] <- 4.5
arthritis$y_merged2 <- factor(arthritis$y_merged2)

fit_merged2 <- osm(y_merged2 ~ baseline + sex + age, data=arthritis,
                   subset = (time == 1), control = list(maxit = 5000))
fit_merged2
#> Call:
#> osm(formula = y_merged2 ~ baseline + sex + age, data = arthritis,
#>     control = list(maxit = 5000), subset = (time == 1))
#>
#> Coefficients beta:
#>     baseline          sex          age
#>  1.202650077  0.605239489 -0.009348869
#>
#> Intercepts mu:
#>       mu1     mu2.5     mu4.5
#>  0.000000  2.001378 -1.719133
#>
#> Score parameters phi:
#>      phi1    phi2.5    phi4.5
#> 0.0000000 0.2202381 1.0000000
#>
#> Residual Deviance: 430.9096
#> AIC: 442.9096
#> BIC: 465.1123
#> (3 observations deleted due to missingness)
```

Recoding the response levels using the score parameters

The other useful aspect of the ϕ_k parameters, apart from an indication of which levels of the response could be merged without much loss of information, is that they can be used to recode the ordinal levels in an empirical way.

That is, if you want to apply numerical methods to the ordinal response variable, then instead of numbering the levels 1, 2, etc. you can number them using the ϕ_k values, which are empirically selected based on the data. If you wish, you can rescale the ϕ_k values so that the levels range between 1 and q, where q is the total number of levels in the response. You simply calculate $v_k = 1 + \phi_k(q-1)$ and then the values v_k will be your new codings for the response levels.

Fernández, et al. (2021) demonstrates a similar approach where the ordered stereotype scores are used as codings for the response levels before applying a numerical method, archetypoid analysis, to the recoded responses. However, as that is a multivariate analysis method, in that particular case the clustering form of the ordered stereotype model was fitted, instead of the regression model.

Note that this recoding is merely a suggestion from the data, not a requirement. If there are good *a priori* reasons for keeping the response levels unmerged, such as wanting to retain consistency with an analysis on another related dataset, then there is no need to merge the levels. The results simply indicate which levels could be safely merged if merging is desirable.

Odds ratios

In the Ordered Stereotype Model, it is not possible to calculate a single odds ratio estimate for each covariate coefficient. This is because the model is purposely constructed so that the covariates can have slightly different effects on the different levels of the response variable. The overall effect of a covariate depends not only on its coefficient but also on the ϕ_k parameter for each level k of the response.

The implication of that is that there is no single number, akin to an odds ratio, that would summarize how the covariate changes the response. We can say which effect sizes are bigger and smaller, and which are positive and negative, but there is no convenient way to summarize the differences caused by a covariate using one single number.

Stereotype Model functions in other packages

Note that the `ordinalgmifs` package also provides a fitting function for the Stereotype Model. However, this package fits the Stereotype Model rather than the Ordered Stereotype Model.

The Ordered Stereotype Model has a restriction $0 = \phi_1 \leq \phi_2 \leq \cdots \leq \phi_q = 1$, where q is the number of levels in the response variable, and this restriction enforces the ordinal nature of the response.

The Stereotype Model, i.e. the model fitted in `ordinalgmifs`, does not have this restriction on the ϕ_k values.

Plotting effects

The `effects` package in R can be used to plot the effects of different covariates. Read the package documentation for detailed information about it:

https://cran.r-project.org/web/packages/effects/index.html

The package has been forked to provide a version modified to work with `clustord`. It still has all the existing functionality of `effects`, so can be used to plot effects for other types of models supported by `effects` (such as `lm`, `glm` and `polr`).

In order to install my version of effects and use it for plotting OSM regression predictor effects, you can use the `remotes` package that provides a function allowing you to install packages directly from GitHub:

```
remotes::install_github("lfmcmillan/effects")
```

If you have tried that, and when you run the model fitting and plot commands e.g.

```
fit1 <- osm(y ~ x + z, data=df)
plot(Effect(focal.predictors = c("x"), fit1))
```

and you still get the following error message, or something similar:

```
Error in glm.fit(x = c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  :
  NA/NaN/Inf in 'y'
In addition: Warning messages:
1: In Ops.ordered(y, mu) : '-' is not meaningful for ordered factors
2: In Ops.ordered(eta, offset) : '-' is not meaningful for ordered factors
3: In Ops.ordered(y, mu) : '-' is not meaningful for ordered factors
```

this means that my version of the effects package has not been installed successfully. In this situation, try running the following code, which will detach the existing version of `effects` from your working R session before trying to install the forked verison:

```
detach("package:effects")
remotes::install_github("lfmcmillan/effects")
```

Then repeat the model fitting and plot Effects commands as before.

The following code would produce an effects plot of the `baseline` predictor:

```
library(effects)
plot(Effect(focal.predictors = c("baseline"), fit))
```

The plot shows the probabilities of getting each level of the response variable, and how those probabilities change across the different values of the `baseline` predictor.

If you have further issues with `clustord` or the forked version of `effects`, please contact me at louise.mcmillan@vuw.ac.nz

References

Fernández, D., Epifanio, I. and McMillan, L. F. (2021) Archetypal analysis for ordinal data. *Information Sciences*, 579, pp. 281–292, https://doi.org/10.1016/j.ins.2021.07.095.