Ordinal regression models made easy. A tutorial on parameter interpretation, data simulation, and power analysis.

Filippo Gambarota[1] & Gianmarco Altoè[1]

[1] Department of Developmental Psychology and Socialization, University of Padova, Italy

Author Note

The authors made the following contributions. Filippo Gambarota: Conceptualization, Writing - Original Draft Preparation, Writing - Review & Editing, Methodology, Software; Gianmarco Altoè: Conceptualization, Writing - Review & Editing, Supervision.

Correspondence concerning this article should be addressed to Filippo Gambarota, Via Venezia 8, 35131, Padova (Italy). E-mail: filippo.gambarota@unipd.it

Abstract

Ordinal data are widespread in psychology such as Likert items, ratings, or generic ordered variables. These variables are usually analyzed using metric models (e.g., standard linear regression) with important drawbacks in terms of statistical inference (reduced power and increased type-1 error) and prediction. One possible reason for not using ordinal regression models could be difficulty in understanding parameters or conducting a power analysis. The tutorial aims to present ordinal regression models using a simulation-based approach. Firstly, we introduced the general model highlighting crucial components and assumptions. Then we explained how to interpret parameters for a logit and probit model. Then we proposed two ways for simulating data as a function of predictors showing a 2x2 interaction with categorical predictors and the interaction between a numeric and categorical predictor. Finally, we showed an example of power analysis using simulations that can be easily extended to complex models with multiple predictors. The tutorial is supported by a collection of custom R functions developed to simulate and understand ordinal regression models. The code to reproduce the proposed simulation, the custom R functions, and additional examples of ordinal regression models can be found on the online Open Science Framework repository (https://osf.io/93h5j).

Keywords: ordinal regression, monte carlo simulations, power analysis

Word count: 6987

Ordinal regression models made easy. A tutorial on parameter interpretation, data simulation, and power analysis.

Introduction

Psychological research make an extensive use of ordinal data. One of the main reason is probably the usage of Likert scales (Likert, 1932). Ordinal data, as defined by Stevens (1946), belongs to a specific type of measurement scale where ordered numbers are assigned to a variable. Beyond Likert-like items, in Psychology there are several applications of ordinal scales. For instance, sociodemographic variables like educational levels or socioeconomic status, as well as general ratings such as pain severity, agreement with a statement, or the evaluation of sensory experiences (e.g., Overgaard & Sandberg, 2021).

In contrast to nominal scales, the labels in ordinal scales are ordered. Unlike interval or ratio scales, there is no explicit assumption about the distance between labels. An example is asking people the degree of agreement about a certain statement using a scale from 1 (no agreement) to 7 (total agreement). Answering 4 compared to 2 suggest an higher agreement but we cannot affirm that there is two times the agreement compared to the second answer. Stevens (1946) and Kemp and Grace (2021) suggested that for ordinal variables is appropriate to calculate ranks-based descriptive statistics (e.g., median or percentiles) instead of metric statistics (e.g., mean or standard deviation) and using appropriate inferential tools (Agresti, 2010; e.g., Cliff, 1996). This distinction in terms of the appropriateness of certain descriptive statistics is also relevant when modeling data. Treating ordinal data as metric refers to assuming the labels as actual integer numbers thus assuming a fixed and know distance between levels (Liddell & Kruschke, 2018).

In Psychology especially with item-based measures (questionnaires, surveys, etc.) the common practice is running a linear regression that makes an explicit assumption about metric features of the response variable. Liddell and Kruschke (2018) reviewed the psychological literature using Likert-based measures and reported how the majority of

papers used metric-based statistical models. In the same work, Liddell and Kruschke (2018) showed examples about the potential pitfalls of metric models applied to ordinal variables (but see Robitzsch, 2020 for an alternative perspective). They reported problems in terms of lack of power, inversion of the effects (e.g., finding a negative effect when the true effect is positive) and biased estimates. While sums or averages of ordinal items could be treated as metric (Carifio & Perla, 2008; Carifio & Perla, 2007; but see Jamieson, 2004) Liddell and Kruschke (2018) provided some examples of potential pitfalls even in this case. In the current paper we discuss only cases where there is a single ordinal outcome (e.g., a single item, question, etc.) without considering the case of aggregating ordinal responses.

For the tutorial, we assume that the reader is familiar with basic R programming and introductory theory about linear regression. Similar to other tutorials (DeBruine & Barr, 2021; Gambarota & Altoè, 2023) we used few equations that are necessary to understand the model parameters and implement the Monte Carlo simulation.

In the first part we introduce the ordinal models explaining the general structure and assumptions. Then we move with model fitting and parameters interpretation. Finally we introduce the simulation approach for common research scenarios and an application to power analysis. We used R [Version 4.3.1; R Core Team (2023)] for the code, figures and tables. Details about R packages an The tutorial is supported by a set of custom R functions that can be found on the online OSF repository (https://osf.io/93h5j). For the tutorial we used the following packages.

```
library(dplyr)
library(tidyr)
library(ordinal)
library(ggplot2)

# custom functions

# show first and last rows of a dataframe
head_tail <- function(x, n = 5){
  rbind(
```

```
    head(x, n = n),
    tail(x, n = n)
  )
}
```

Ordinal regression models

We can generally define as *ordinal regression*, a statistical model explicitly considering the ordinal (and not metric) nature of the response variable. The nomenclature of *ordinal regression* models can be confusing but Tutz (2022) and Bürkner and Vuorre (2019) provide a clear and updated taxonomy of ordinal regression models.

We can identify three main models: *cumulative models* (CM, Agresti, 2010; McCullagh, 1980), *sequential models* (Tutz, 1990), and *adjacent category models*. Among these, the CM is the most widely used, assuming the existence of a latent variable categorized using a set of thresholds. The *sequential model* is appropriate when modeling sequential processes where responding to a certain category implies have already reached lower categories. The *adjacent category model* compares a category with the successive one and can be seen the adjacent-category model can be seen as a series of binomial regressions taking into account the ordering (Tutz, 2022).

In the current paper we focus on the CM for several reasons. Firstly, the latent formulation of the model is particularly convenient for parameter interpretation and data simulation. The second reason is that several psychological construct can be formalized as a latent continuous variable observed as an ordinal item(s). Furthermore, the signal detection theory framework that is very common in experimental psychology can be implemented using a CM (e.g., DeCarlo, 2010). Figure 1 depicts the overall structure of the CM.

Model notation

In this section we introduce some notation for the CM to be consistent with the existing literature and the `ordinal` R package. We define Y_k as the observed ordinal variable with k levels and $Y^\star$ is the underlying latent variable. The latent variable is segmented using $k-1$ thresholds $\alpha_1, \dots, \alpha_{k-1}$. Similarly to the generalized linear models framework (e.g., Fox, 2015), we define $g(p) = \eta$ as the link function that maps probabilities into the linear predictor η. To transform back η into probabilities we use the inverse of the link function $p = g^{-1}(\eta)$. The specific link function define the type of model and require different R functions. For example, using a Normal distribution (*probit* link function) requires using the cumulative distribution function $g(p) = \Phi^{-1}(p) = \eta$ and the inverse of the link function is the inverse cumulative distribution function (or quantile function) defined $p = g^{-1}(\eta) = \Phi(\eta)$.

When modelling an ordinal variable in a CM we include predictors on the cumulative probability $P(Y \leq k), k = 1, \dots, k-1$[1]. Equation (1) shows the general cumulative model with predictors $\mathbf{X}$ and coefficients β. The minus sign in $\mathbf{X}\beta$ is used to interpret the β_j as in the standard regression models where higher positive values corresponds to increased probability of responding higher k categories (Agresti, 2010).

$$P(Y \leq k) = g^{-1}(\alpha_k - \mathbf{X}\beta) \;\; k = 1, \dots, k-1 \tag{1}$$

The $\mathbf{X}\beta$ is the linear predictor η that is the cumulative probability $P(Y \leq k)$ transformed using the link function $g(\cdot)$. The probability of a single outcome $P(Y = k)$ can be calculated as the difference between cumulative probabilities (see Equation (2)).

$$P(Y = k) = g^{-1}(\alpha_k - \eta) - g^{-1}(\alpha_{k-1} - \eta), \;\; k = 1, \dots, k-1 \tag{2}$$

[1] As done by Agresti (2010), when referring to $P(Y \leq k)$ we are implicitly conditioning on a particular $\mathbf{X}$ value $P(Y \leq k|\mathbf{X}_i)$

There are always two special cases when computing the probability of a single outcome Y that is when $Y = 1$ and $Y = k$. In the first case the cumulative probability is calculated from $-\infty$ (assuming that $\alpha_0 = -\infty$). In the second case ($Y = 1$) the probability is calculated as $P(Y = k) = 1 - g^{-1}(\alpha_{k-1} - \eta)$ (a $\alpha_k = +\infty$. The Figure 1 shows how the single probabilities of the ordinal outcome are calculated from cumulative probabilities.

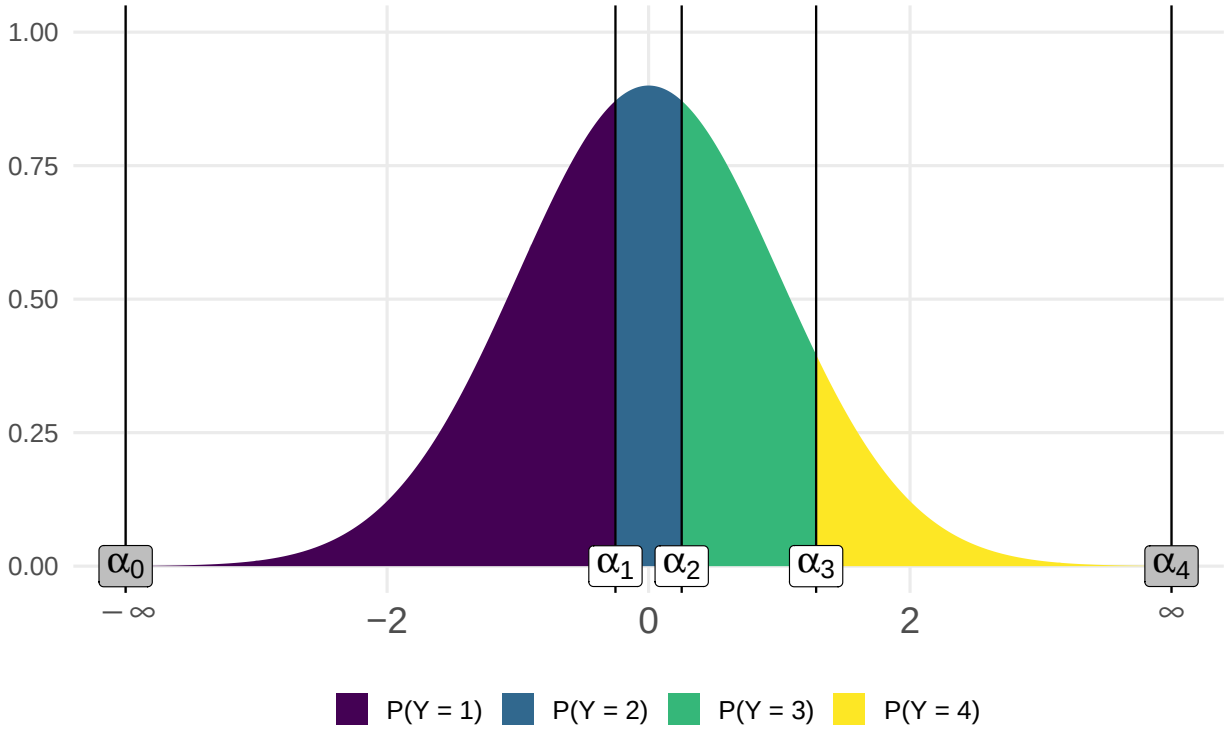

Figure 1. Relationship between cumulative probabilities and ordinal outcomes. The $Y^\star$ latent variable is segmented into k levels using $k - 1$ thresholds (α). The distance between thresholds determine the probability of each outcome. The grey thresholds are only used to compute the $Y = 1$ and $Y = k$ probabilities.

[Figure 1 about here]

Equation (3) shows the latent formulation of the previous model. The latent variable $Y^\star$ is a function of the linear predictor $\eta = \mathbf{X}\beta$ similar to a standard regression.

$$\mathbf{Y}^{\star} = \mathbf{X}\beta + \epsilon \tag{3}$$

The ϵ term is the random part (errors) of the model. For a *probit* model, errors are distributed as a standard Normal distribution and for a *logit* model as a standard logistic distribution. Following the notation by Tutz (2022), the observed ordinal value $Y_i = k$ comes from $Y_i^{\star}$ belonging to the interval defined by the thresholds $Y_i = k \iff \alpha_{k-1} < Y_i^{\star} < \alpha_k$ where $-\infty = \alpha_0 < \alpha_1 < \cdots < \alpha_{k-1} < \alpha_k = \infty$.

In the CM, thresholds α_k are considered fixed and part of the measurement procedure (Liddell & Kruschke, 2018; but see the *location-shift* models by Tutz, 2022, where thresholds vary as a function of predictors) and can be considered as intercepts. The proposed CM can be parametrized in different ways. Liddell and Kruschke (2018) proposed a Bayesian version of the model and Gelman, Hill, and Vehtari (2020) proposed other thresholds parametrizations.

Link function

The CM implemented in Equations (1) and (3) requires specifying the link function $g(\cdot)$ or the errors distribution $\epsilon_i \sim D(\mu, \sigma^2)$ (D being a certain probability distribution). Among several available functions the *logit* and *probit* models are the most common. The *logit* model use a *logit* link function and a standard logistic distribution as latent variable. On the other side, the *probit* model assume a standard Normal distribution using the inverse of the cumulative distribution ($\Phi^{-1}(\cdot)$) function as link function.

The two models provide similar results with a different parameters interpretation. In the next sections we will illustrate the differences and simulation strategies. Figure 2 depicts the two distributions while Table 1 summarise the presented CM with the proposed link function and the corresponding R code.

In terms of parameters, both distributions can be defined with in terms of location μ

and scale s. The standard normal distribution has $\mu = 0$ and $s = 1$. Furthermore the variance corresponds to the scale $s^2 = \sigma^2 = 1$. The variance of the logistic distribution is $\sigma^2 = \frac{s^2\pi^2}{3}$. The standard logistic distribution has $\mu = 0$ and $s^2 = 1$ thus the standard deviation simplified to $\frac{\pi}{\sqrt{3}} \approx 1.81$. In practical terms, fixing μ and s lead to an higher standard deviation for the logistic distribution. In the supplementary materials we showed how to implement a location-scale models for including predictors on the scale parameter (Cox, 1995; Rigby & Stasinopoulos, 2005; Tutz, 2022).

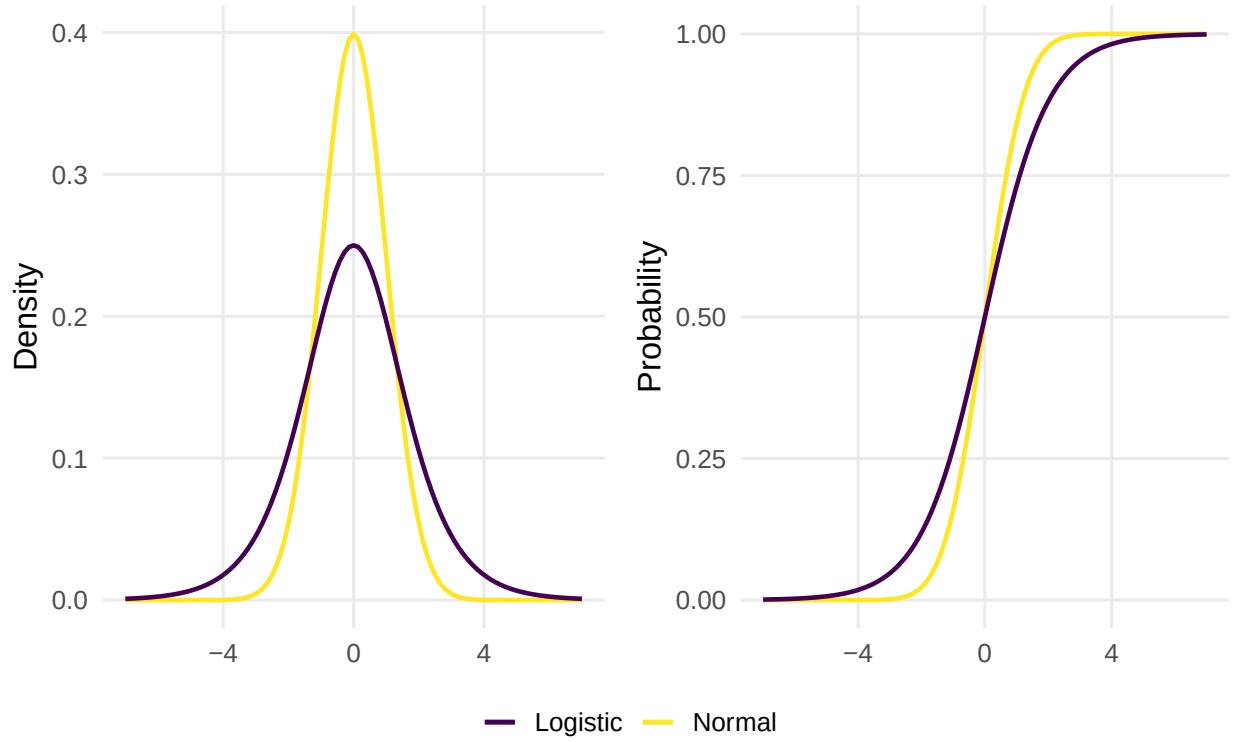

Figure 2. Difference between *logit* and *probit* models. On the left the probability density function (PDF). On the right the cumulative distribution function (CDF).

[Figure 2 about here]

Model fitting

For fitting the CM we used the `ordinal` package (Christensen, 2023). The `ordinal` package provide a complete and intuitive way to implement the ordinal models. The

Table 1

Summary of the probit and logit models in terms of link $g(\cdot)$ and inverse link function $g^{-1}(\cdot)$ and the corresponding R code.

	Link Function $g(.)$		Inverse Link Function $g^{-1}(.)$	
Model	**Equation**	**R Code**	**Equation**	**R Code**
Cumulative Logit	$\text{logit}(p) = \log(p/(1-p))$	`qlogis()`	$e^{\text{logit}(p)}/(1+e^{\text{logit}(p)})$	`plogis()`
Cumulative Probit	$z = \Phi^{-1}(p)$	`qnorm()`	$p = \Phi(z)$	`pnorm()`

syntax is very similar to standard linear models in R with functions to calculate predictions, perform model comparison and extract relevant model information.[2]

The main function is `clm()` and the formula is specified using the syntax `y ~ ...` where `y` is the ordinal response and `...` is a combination of predictors. The package also implements mixed-effects models including random intercepts and slopes.

When fitting the model the crucial arguments are the `formula`, the `link` function and the `data`. More advanced arguments are `nominal`, `scale` and `threshold`.

- `formula`: the formula `y ~ x` with the dependent variable and predictors.
- `link`: is the link function (in this tutorial only *logit* and *probit*)
- `data`: is the dataset with variables included in the `formula`
- `nominal`: formula with predictors where the proportional odds assumption (See Section) is relaxed (i.e., partial or non proportional odds)
- `scale`: formula with predictors for the scale parameter for fitting a scale-location model

[2] For a very complete overview of the ordinal package see https://cran.r-project.org/web/packages/ordinal/ordinal.pdf and https://cran.uni-muenster.de/web/packages/ordinal/vignettes/clm_article.pdf

- `threshold`: different structures for estimating the thresholds. The default is `threshold = "flexible"` where $k-1$ threshold (where k is the number of ordinal levels for Y) are estimated.

We can start by fitting a simple model, highlighting the crucial parameters where the detailed explanation will be expanded in the next sections. Table 2 contains simulated data from $n = 5000$ large n for having stable estimations in the examples) participants rating the agreement about an item with $k = 4$ ordered options. The participants are divided into two groups (X_a and X_b). We can fit a cumulative link model (`clm()` function) and check the model summary.

Table 2

Summary of the simulated dataset. For each group (a and b) we reported mean, median and standard deviation of the ordinal outcome (metric descriptive statistics). For each ordinal outcome (Y) we reported the frequency and (within parentheses) the probability.

Group	Mean	Median	SD	Y1	Y2	Y3	Y4
a	2.5	2	1.1	**599** (p = *11.98*)	**662** (p = *13.24*)	**627** (p = *12.54*)	**612** (p = *12.24*)
b	3.17	3.5	0.99	**221** (p = *4.42*)	**381** (p = *7.62*)	**648** (p = *12.96*)	**1250** (p = *25*)

```
fit <- clm(y ~ x, data = dat, link = "logit")
summary(fit)
#> formula: y ~ x
#> data:    dat
#>
#>  link  threshold nobs logLik   AIC      niter max.grad cond.H
#>  logit flexible  5000 -6458.45 12924.89 4(0)  1.59e-08 1.7e+01
#>
#> Coefficients:
#>    Estimate Std. Error z value Pr(>|z|)
#> xb  1.14864    0.05333   21.54   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
```

```
#> Threshold coefficients:
#>     Estimate Std. Error z value
#> 1|2 -1.16300    0.04345 -26.765
#> 2|3  0.01284    0.03835   0.335
#> 3|4  1.14143    0.04194  27.214
```

The two main sections of the model summary are the *Coefficients* section reporting the regression coefficients β and the *Threshold* section reporting the α estimation. Given that $k = 4$ we have $k - 1 = 3$ thresholds and one β associated with the x effect. As in standard regression models, when x is a categorical predictor with j level, we will estimate $j - 1$ regression coefficients (plus the intercept term) where the interpretation depends on the contrast coding (see Schad, Vasishth, Hohenstein, & Kliegl, 2020). In R the default is the dummy coding where a factor of j levels is converted into $j - 1$ dummy variables. By default, the first level of the factor is taken as the reference level and the $j - 1$ coefficients represent the comparison between the other levels and the reference.

Interpreting parameters

Logit Model. Odds and odds ratio

To understand the logit model we need to introduce odds and odds ratio. The odds of a probability p is defined as $p/(1 - p)$ thus the success probability over the failure probability. The odds takes value ranging from 0 to ∞. With a probability of $p = 0.8$, we have odds of 4, indicating that there are four successes for each failure. The same as having $p = 0.2$ and an odds of 0.25 means that for each 0.25 successes we have a failure (or 4 failures for each success). When comparing two groups or conditions we can take the ratio of two odds. An odds ratio (OR) of 4 means that the odds of success at the numerator is 4 times higher than the odds of success at the denominator. The Figure 3 shows the relationship between probabilities and odds. The logit transformation is about taking the

logarithm of the odds creating a symmetric function ranging from $-\infty$ to ∞ with $p = 0.5$ as the midpoint because $\log(0.5/(1-0.5)) = 0$.

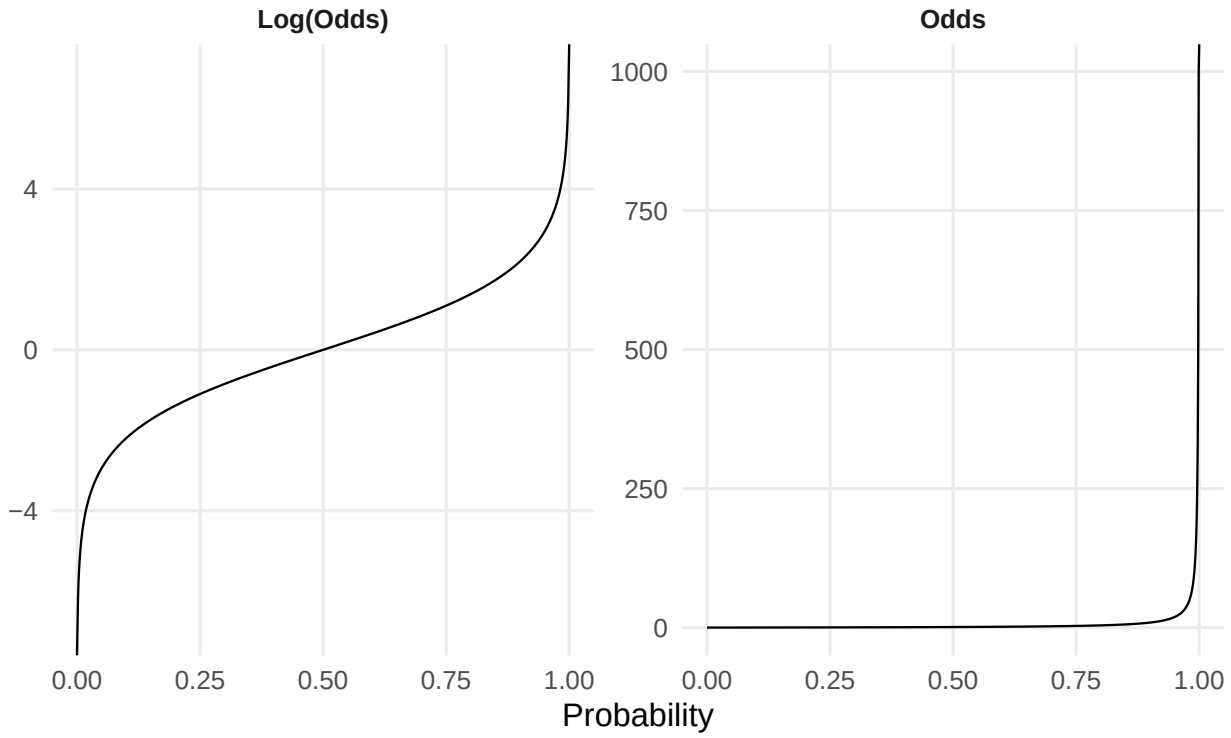

Figure 3. Relationship between probability (x axis) odds and the log odds. Using log odds ensure a symmetric relationship with a zero midpoint ($p = 0.5$).

[Figure 3 about here]

Odds and odds ratios are clearly defined with $k = 2$ outcomes. With an ordinal variable we have $k - 1$ odds ratios determined by the cumulative probability in terms of $P(Y \leq 1), \ldots, P(Y \leq k-1)$ comparing the two groups.

For example the probability of responding $Y \leq 1$ in the group "a" is 0.24 corresponding to an odds of 0.32. When calculating the odds ratio comparing "a" vs "b" for $Y \leq 1$ we obtain that the group "a" has 3.25 times the odds of responding $Y \leq 1$ compared to the group "b".

```r
# the dummy_ord() is a custom function creating k - 1 dummy variable for the cumulative probabilities

odds <- function(p) p / (1 - p)

# data.frame with the group (x) and the k - 1 dummy variables
cum_p <- cbind(x = dat$x, dummy_ord(dat$y))

# calculating the cumulative probability for k - 1 variables.
# Taking the average of a series of 0-1 is the same as computing the proportion of 1s.
(group_a <- apply(cum_p[cum_p$x == "a", -1], 2, mean))
#> y1vs234 y12vs34 y123vs4
#>  0.2396  0.5044  0.7552
(group_b <- apply(cum_p[cum_p$x == "b", -1], 2, mean))
#> y1vs234 y12vs34 y123vs4
#>  0.0884  0.2408  0.5000

# calculating k - 1 odds ratios on the cumulative probabilities as a/b
(ors <- odds(group_a) / odds(group_b))
#>  y1vs234  y12vs34  y123vs4
#> 3.249352 3.208806 3.084967
log(ors) # logarithm of the odds
#>  y1vs234  y12vs34  y123vs4
#> 1.178456 1.165899 1.126541
```

Proportional odds assumption

The ORs calculated above are very similar to the parameter estimated from the model (thanks to the large n). The CM estimate essentially the (cumulative) OR as a function of predictors. Furthermore, the $k-1$ ORs are very similar because data are generated under the so-called proportional odds (PO) assumption. The basic version of the CM model fitted above assume the PO estimating a single β (instead of $k-1$). The PO assumption is formalized in Equation (4). The cumulative log odds ratio comparing $P(Y_k|x_0)$ with $P(Y_k|x_1)$ (x_0 and x_1 being two levels of a predictor) is the same regardless the specific threshold ($\beta_1 = \beta_2 \cdots = \beta_{k-1}$). The Figure (4) depicts the proportional odds assumption for the $k-1$ logistic curves both for probabilities and linear predictors η.

$$\text{logit}(\frac{P(Y \leq 1|x_1)}{P(Y \leq 1|x_0)}) = \cdots = \text{logit}(\frac{P(Y \leq k-1|x_1)}{P(Y \leq k-1|x_0)}) \tag{4}$$

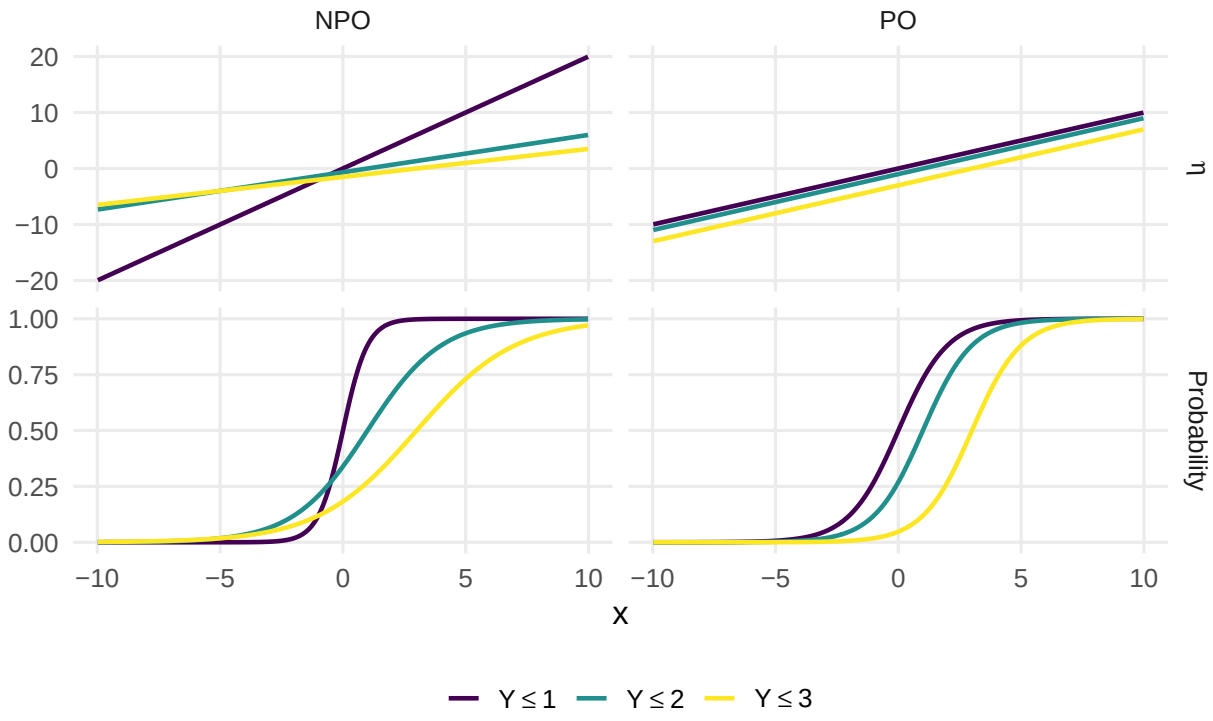

Figure 4. Example of proportional odds (PO, right) and non-proportional odds (NPO, left) for cumulative probabilities and the linear predictor η assuming a continuous predictor x. Data supporting the PO assumption shows an horizontal shift in the cumulative probability with the same slope while for NPO model slopes can be heterogeneous.

[Figure 4 about here]

There are CM models relaxing this assumption completely (*non proportional odds*, Tutz, 2022) or partially (*partial proportional odds*, Peterson & Harrell, 1990). The advantage of the PO model is the parsimony. In some scenarios, the PO assumption can be considered too strict. There are several methods for testing if data are supporting the PO (see Liu, He, Tu, & Tang, 2023 for an overview). Tutz and Berger (2020) suggested a trade-off between assuming/relaxing the PO assumption by fitting *location-shift* or

location-scale models. Basically these methods should guarantee more flexibility in modelling the observed probabilities reducing the number of parameters. However, these methods and models are outside the scope of the tutorial.

The PO can also be clearly seen calculating the ORs using predicted probabilities. By default, the model fitted with `clm()` assume the PO. In the following code block we computed the odds ratio comparing x_a and x_b when $k \leq 1$ and $k \leq 2$.

```
# fitting the model
fit <- clm(y ~ x, data = dat, link = "logit")

# extracting the predicted probabilities for the two groups
pr <- predict(fit, data.frame(x = unique(dat$x)))$fit

# y <= 1
y1a <- pr[1, 1]
y1b <- pr[2, 1]

# y <= 2
y12a <- sum(pr[1, 1:2])
y12b <- sum(pr[2, 1:2])

# odds ratio y <= 1 (a vs b) VS y <= 2 (a vs b)
odds(y1a) / odds(y1b)
#> [1] 3.153894
odds(y12a) / odds(y12b)
#> [1] 3.153894
```

Probit Model and z scores

The previous model was fitted using a *logit* link function. When assuming a standard normal distribution we are fitting *probit* model. The main difference regards the parameters interpretation. In the logit model the β are the log odds ratio. For categorical variables they represents the increase in the log odds of moving from one category to another while for numerical variables is the increase in the log odds for a unit increase in x.

In the *probit* model, the β is the increase in terms of z scores for a unit increase in x. This is very convenient especially for categorical variables because parameters can be interpreted as a Cohen's d like measure. In terms of latent distributions, the β is the shift in the latent mean comparing two or more groups or the slope of latent scores as a function of a numeric x. More formally the shift in the latent distribution is β/σ (for the *probit* model $\sigma = 1$). The interpretation in terms of shifting the latent mean holds also for the logistic model. However, the standard deviation of the standard logistic regression is $\frac{\pi}{\sqrt{3}} \approx 1.81$[3]. The β for the logistic distribution can be interpreted as the location shift of the latent logistic distribution by $\beta/(\frac{\pi}{\sqrt{3}})$ standard deviations (Agresti, 2010).

Proportional odds and *probit* model. More generally, the PO assumption can be called *parallel slopes*. When applying the link function, cumulative probabilities have no longer a sigmoid shape but linear (see Figure 4). If the steepness of the sigmoid (i.e., β_j) is not constant the $k-1$ lines have different slopes. The PO assumption is relevant only for the *logit* model because parameters are ORs.

For the *probit* model, the difference in z scores for a unit increase in x is the same regardless of the threshold (see Equation (5). As for the previous example, we can fit the CM and manually calculate the difference in z scores (from predicted probabilities).

$$z_{x_1 - x_0} = \Phi(P \leq 1|x_1) - \Phi(P \leq 1|x_0) = \cdots = \Phi(P \leq k-1|x_1) - \Phi(P \leq k-1|x_0) \quad (5)$$

```r
# fitting the model
fit <- clm(y ~ x, data = dat, link = "probit")


# extracting the predicted probabilities for the two groups
pr <- predict(fit, data.frame(x = unique(dat$x)))$fit
```

[3] Actually the variance of the logistic distribution is $\frac{s^2\pi^2}{3}$ and the standard deviation $\frac{s\pi}{\sqrt{3}}$ where s is the scale of the distribution. For the standard logistic distribution $s = 1$ (as for the standard normal distribution).

```r
# y <= 1
y1a <- pr[1, 1]
y1b <- pr[2, 1]

# y <= 2
y12a <- sum(pr[1, 1:2])
y12b <- sum(pr[2, 1:2])

# z score difference y <= 1 (a vs b) VS y <= 2 (a vs b)
qnorm(y1a) - qnorm(y1b)
#> [1] 0.6869165
qnorm(y12a) - qnorm(y12b)
#> [1] 0.6869165
```

Simulating data

In this tutorial we present two method for simulating ordinal data. Simulating data is a powerful strategy to understand the model (DeBruine & Barr, 2021) and estimate statistical properties (e.g., power or type-1 error). The first simulation method calculate the probabilities of each Y level as a function of predictors and generate data from a *multinomial* distribution. The second method simulate data using the the latent formulation. Whenever random number generation occurs, it is appropriate to set a `seed` (`set.seed()` function, in this case we use `set.seed(2024)`). Running again the same code with the same seed will produce the same result. A general simulation approach concerns generating a dataset from the assumed data generation process, fitting the statistical model and assessing the recovery of simulated parameters. To check the simulation approach, using a large sample size produce estimations with small sampling error. A limited sample size even when fitting the true model will produce variable estimations.

Simulating from a multinomial distribution. For the first method we need to calculate $P(Y \leq k) = g^{-1}(\eta)$ as a function of predictors and then sample from a *multinomial* (more specifically *categorical*) distribution using the `sample()` function in R. This method is similar to the general way of simulating data for a generalized linear model

(see the supplementary materials).

As a simple example, we simulate two groups with $n = 100$ participants responding to an item (Y) with $k = 4$ ordered options. As explained in the previous sections, we can summarise the effect size of a CM (assuming PO) using a single $\beta = \log(OR) = \log(3)$. For simplicity, the probabilities of the first group $x = 0$ are uniform thus $P(Y = 1|x_0) = ... P(Y = k|x_0) = 1/k$. The following code summarise the first steps of the simulation. We define the simulation parameters, calculate the $k - 1$ thresholds α and apply Equations (1) and (2) calculating the probability of each response.

To set the thresholds, we can consider them as intercepts thus the transformed probabilities of each ordinal response when $x = 0$ (in this case the first group). The `alpha_to_prob()` and `prob_to_alpha()` can be used to go back and forth from thresholds and probabilities.

```
k <- 5
(p <- rep(1/k, k)) # uniform
#> [1] 0.2 0.2 0.2 0.2 0.2
names(p) <- paste0("y", 1:k)

(alpha <- prob_to_alpha(p, link = "logit")) # or prob_to_alpha(p, "probit")
#>        1|2        2|3        3|4        4|5
#> -1.3862944 -0.4054651  0.4054651  1.3862944
alpha_to_prob(alpha, link = "logit")
#>  p1  p2  p3  p4  p5
#> 0.2 0.2 0.2 0.2 0.2
```

```
## SIMULATION PARAMETERS

N <- 100 # sample size
or <- 3 # odds ratio
k <- 4 # number of ordinal alternatives
prob0 <- rep(1/k, k) # probabilities for the first group
alpha <- prob_to_alpha(prob0, link = "logit")
dat <- data.frame(x = rep(c(0, 1), each = N/2))

## LINEAR PREDICTOR
```

```r
# calculate linear predictor using equation 1 obtaining k - 1 equations

lp <- lapply(alpha, function(a) a - log(or) * dat$x)
names(lp) <- sprintf("lp_leq%s", 1:(k - 1)) # giving appropriate names
lp <- data.frame(lp)
head_tail(lp, n = 3)
#>       lp_leq1   lp_leq2  lp_leq3
#> 1   -1.098612  0.000000 1.098612
#> 2   -1.098612  0.000000 1.098612
#> 3   -1.098612  0.000000 1.098612
#> 98  -2.197225 -1.098612 0.000000
#> 99  -2.197225 -1.098612 0.000000
#> 100 -2.197225 -1.098612 0.000000

## CUMULATIVE PROBABILITIES

# apply the inverse of the link function (invlogit) to calculate cumulative probabilities
cump <- lapply(lp, plogis)
cump <- data.frame(cump)

# giving appropriate names, cump = cumulative probability, leq = less or equal
names(cump) <- sprintf("cump_leq%s", 1:(k - 1))
head_tail(cump, n = 3)
#>     cump_leq1 cump_leq2 cump_leq3
#> 1        0.25      0.50      0.75
#> 2        0.25      0.50      0.75
#> 3        0.25      0.50      0.75
#> 98       0.10      0.25      0.50
#> 99       0.10      0.25      0.50
#> 100      0.10      0.25      0.50

## PROBABILITIES OF Y

# for each row, we can calculate P(Y = k) using equation 2
# P(Y = 1) = P(Y <= 1)
# P(Y = 2) = P(Y <= 2) - P(Y <= 1)
# P(Y = 3) = P(Y <= 3) - P(Y <= 2)
# P(Y = 4) = 1 - P(Y <= 3)

# adding a columns of 0 and 1, then diff() for adjacent differences
```

```r
cump <- cbind(0, cump, 1)
p <- apply(cump, 1, diff, simplify = FALSE)

p <- data.frame(do.call(rbind, p)) # collapse list of rows into a dataframe
names(p) <- sprintf("p%s", 1:k) # giving appropriate names

# probabilities for id = 1 (x = 0) and id = 51 (x = 1)
p[c(1, 51), ]
#>      p1   p2   p3   p4
#> 1  0.25 0.25 0.25 0.25
#> 51 0.10 0.15 0.25 0.50

## CUMULATIVE ODDS RATIO

# calculate the (cumulative) odds ratio

x0 <- cump[1, 2:k]
x1 <- cump[51, 2:k]

# this is the same as the or (the b1) and we are assuming POA
odds(x0) / odds(x1)
#>   cump_leq1 cump_leq2 cump_leq3
#> 1         3         3         3
```

```r
## SAMPLING FROM THE MULTINOMIAL DISTRIBUTION

set.seed(2024)

# example of n random Y outcome based on the probabilities
sample(x = 1:k, size = 10, prob = p[1, ], replace = TRUE)
#>  [1] 1 3 4 4 3 4 3 3 1 2

# we can apply it to the full dataset, this step lead to different results
# each time we run the code because we are sampling from a distribution

dat$y <- apply(p, 1, function(ps) sample(1:k, size = 1, prob = ps))
head_tail(dat, n = 3)
#>     x y
#> 1   0 1
#> 2   0 1
```

```
#> 3   0 4
#> 98  1 4
#> 99  1 4
#> 100 1 2

# let's compute the observed probabilities, to be compared to the true
# probabilities

# observed
(op <- prop.table(table(dat$x, dat$y), margin = 1))
#>
#>        1    2    3    4
#>   0 0.26 0.24 0.26 0.24
#>   1 0.06 0.10 0.30 0.54

# true (just selecting a row from x = 0 and x = 1)
p[c(1, 51), ]
#>      p1   p2   p3   p4
#> 1  0.25 0.25 0.25 0.25
#> 51 0.10 0.15 0.25 0.50

# similarly we can compute the observed cumulative odds ratios

(cum_op <- apply(op, 1, cumsum))
#>
#>        0    1
#>   1 0.26 0.06
#>   2 0.50 0.16
#>   3 0.76 0.46
#>   4 1.00 1.00
odds(cum_op[-k, 1]) / odds(cum_op[-k, 2]) # -k remove the P(y = k) = 1
#>        1        2        3
#> 5.504505 5.250000 3.717391

# the odds ratios are not the same as the parameter. as we increase N
# the parameter will converge to the true value
```

```
dat$y <- ordered(dat$y) # make an ordered factor in R where 1 < 2 < 3 < 4
fit <- clm(y ~ x, data = dat, link = "probit")
summary(fit)
#> formula: y ~ x
```

```
#> data:    dat
#>
#>  link   threshold nobs logLik  AIC    niter max.grad cond.H
#>  probit flexible  100  -124.19 256.38 5(0)  5.62e-08 1.5e+01
#>
#> Coefficients:
#>   Estimate Std. Error z value Pr(>|z|)
#> x   0.8831     0.2276    3.88 0.000104 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Threshold coefficients:
#>     Estimate Std. Error z value
#> 1|2 -0.65508    0.17861  -3.668
#> 2|3 -0.04056    0.16814  -0.241
#> 3|4  0.75376    0.17871   4.218
```

We can generalize the previous workflow as:

1. Define simulation parameters, intercept probabilities (`prob0`), sample size, etc.
2. Define regression coefficients β and calculate the $k-1$ linear predictors (η) using the equations
3. Apply the inverse of the link function $g^{-1}(\eta)$ on the linear predictor and calculate the cumulative probabilities $p(y \leq 1|x), p(y \leq 2|x), ...p(y \leq k-1|x)$.
4. Calculate the probabilities of k outcomes
5. Sample n outcomes from a *multinomial* distribution using the calculated probabilities
6. Fit the appropriate model using `ordinal::clm()`

Simulating from the latent distribution. More efficiently, we can use the latent formulation of the CM. This require simulating a standard linear regression using the appropriate data generation function (*logistic* or *normal*) and cutting the latent variable using the thresholds. The workflow is slightly different compared to the previous approach.

1. Define simulation parameters as in the previous simulation. `prob0` are the probabilities when all predictors $\mathbf{X}$ are zero.
2. Define regression coefficients β and calculate the linear predictor η using the Equation (3)
3. Add the random errors ϵ_i sampling from the logistic or normal distributions
4. Cut the latent variable into k areas using the thresholds α and assign the corresponding ordinal value. This can be done using the `cut()` or the `findInterval()` functions.

The Figure 5 depicts the simulated Y^* and the corresponding ordinal value. As for the previous simulation we can fit the model using `clm()` and check the estimated parameters.

```
## SIMULATION PARAMETERS

set.seed(2024)
N <- 1e3 # sample size
or <- 4 # odds ratio, higher here just for a more clear plot
k <- 4 # number of ordinal alternatives
probs0 <- rep(1/k, k) # probabilities for the first group
alpha <- prob_to_alpha(probs0, link = "logit") # thresholds
dat <- data.frame(x = rep(c(0, 1), each = N/2))

## LINEAR PREDICTOR

# calculate the linear predictor using the model equation
dat$lp <- log(or) * dat$x

# add the random part by sampling errors from a standard logistic (or normal) distribution
dat$ystar <- dat$lp + rlogis(N, location = 0, scale = 1)

# cut the latent distribution. The + 1 because the first category is 0 by default.
dat$y <- findInterval(dat$ystar, alpha) + 1

head_tail(dat, n = 3)
#>      x       lp      ystar y
```

```
#>    0 0.000000  1.6356528 4
#>    0 0.000000 -0.7497878 2
#>    0 0.000000  0.7554420 3
#>  998 1.386294  2.2813322 4
#>  999 1.386294  2.2839112 4
#> 1000 1.386294  2.0325030 4
```

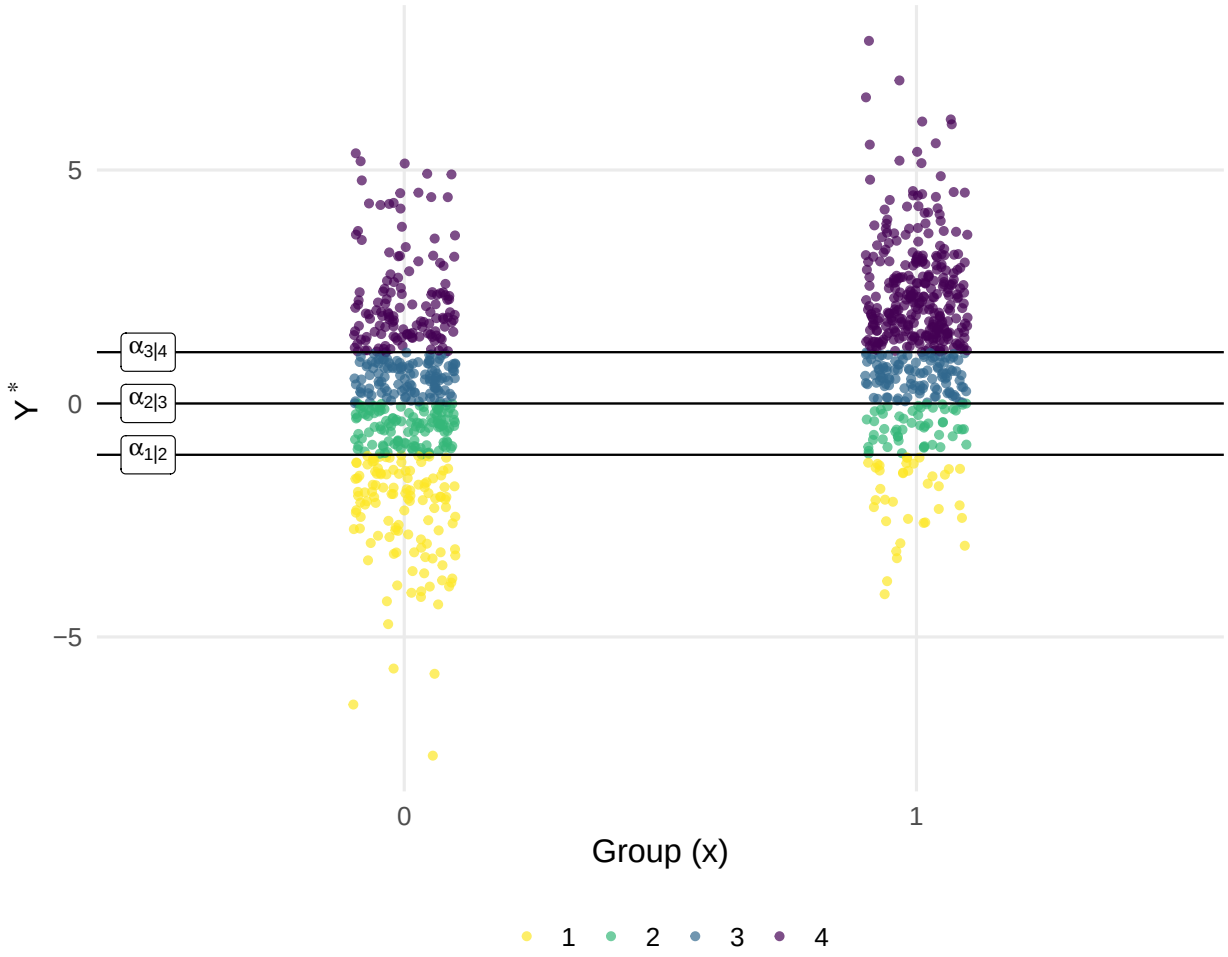

Figure 5. Example of cutting a latent variable for $k = 4$ ordinal outcomes with the effect of a binary predictor x. The thresholds are fixed and the difference between the latent means of the two groups increase the number of points (thus probability) in higher outcomes.

[Figure 5 about here]

```r
dat$y <- ordered(dat$y) # make an ordered factor in R where 1 < 2 < 3 < 4
fit <- clm(y ~ x, data = dat, link = "logit")
summary(fit)
#> formula: y ~ x
#> data:    dat
#>
#>  link  threshold nobs logLik   AIC     niter max.grad cond.H
#>  logit flexible  1000 -1226.66 2461.32 4(0)  2.25e-07 1.7e+01
#>
#> Coefficients:
#>   Estimate Std. Error z value Pr(>|z|)
#> x   1.4078     0.1233   11.41   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Threshold coefficients:
#>     Estimate Std. Error z value
#> 1|2 -1.13646    0.09757 -11.648
#> 2|3 -0.07934    0.08603  -0.922
#> 3|4  0.98793    0.09269  10.658
```

The simulation using the latent formulation of the model is implemented in the `sim_ord_latent()` function. Basically, we define the dataset `dat` with predictors. Then the model formula is specified within the function as `location = ~` along with the vector of regression coefficients (`beta`), baseline probabilities (`prob0`) and the link function. The function return a dataset with the simulated Y and the latent variable (this is possible only because we simulateds the data, otherwise $Y^\star$ cannot be observed by definition).

```r
set.seed(2024)
N <- 100 # sample size
or <- 4 # odds ratio, higher here just for a more clear plot
k <- 4 # number of ordinal alternatives
probs0 <- rep(1/k, k) # probabilities for the first group
alpha <- prob_to_alpha(probs0, link = "logit")
dat <- data.frame(x = rep(c(0, 1), each = N/2))


# same as the previous simulation
dat <- sim_ord_latent(location = ~x, beta = log(or), prob0 = probs0, data = dat, link = "logit")
```

```
head_tail(dat, n = 3)
#>     x y         ys
#> 1   0 4  1.6356528
#> 2   0 2 -0.7497878
#> 3   0 3  0.7554420
#> 98  1 4  1.4134548
#> 99  1 4  5.0032143
#> 100 1 4  1.6703959
```

Choosing parameters values.

Thresholds α***.*** The previous simulation can be easily extended by adding more predictors and their interactions. The crucial part is setting appropriate and empirically meaningful parameters. Thresholds are the quantiles of the latent distribution that produced certain k probabilities when $\mathbf{X} = 0$. To set meaningful α values we can convert probabilities into thresholds (`alpha_to_prob()`). The function `show_alpha()` produce a meaningful visual representation of using a specific set of thresholds (see Figure 6). Thus with two groups for example, the thresholds are the k probabilities of the ordinal variable Y for the reference group (when $x = 0$).

[Figure 6 about here]

Regression coefficients. For *probit* models we can set the β_j to be in standardized (i.e., Cohen's d-like) units. For a categorical variable as the previous example with the group, β_j is the degree of separation in standard deviation unit between the two latent distributions. For *logit* models we can set the odds ratio. Meaningful odds ratios can be derived from previous literature, meta-analyses or converting from other effect sizes. For example, Sánchez-Meca, Marín-Martínez, and Chacón-Moscoso (2003) proposed some equations to convert between odds ratios and Cohen's d. Using their approach, a Cohen's $d = 0.5$ usually considered a plausible medium effect size corresponds to and odds ratio of ≈ 2.47.

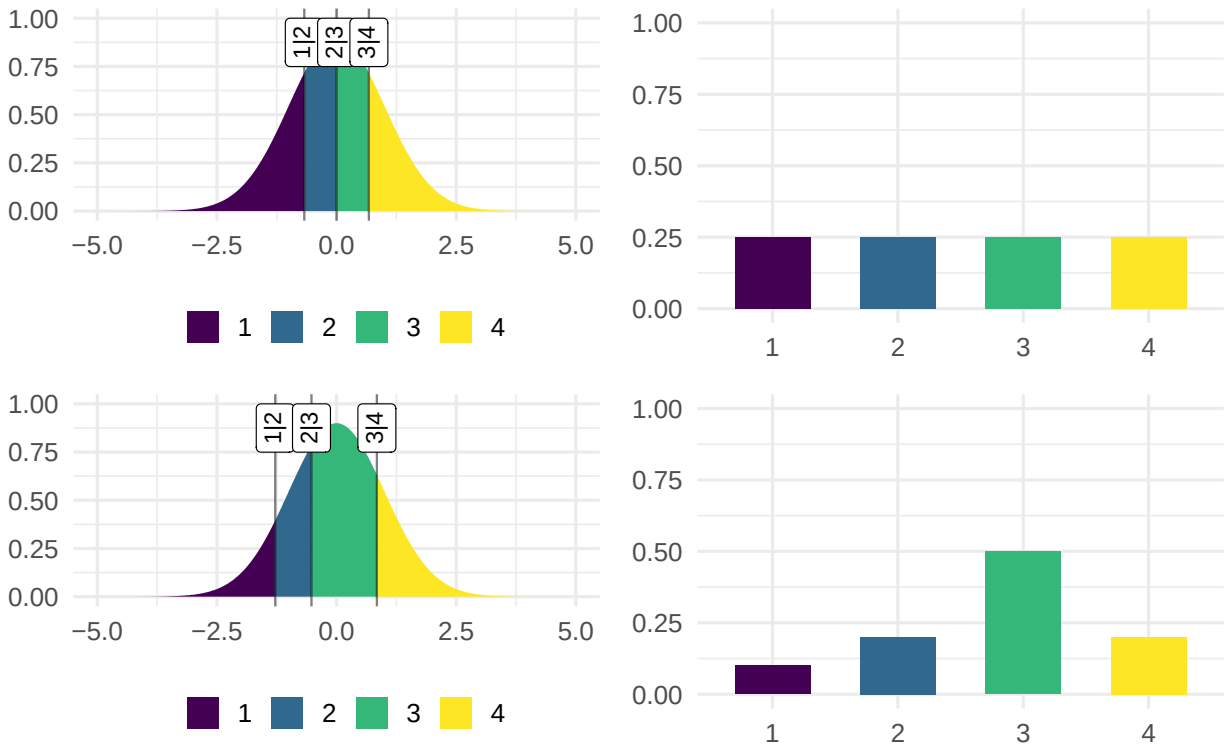

Figure 6. Example of the `show_alpha()` function. On the left the assumed latent variable and $k-1$ thresholds with the corresponding probabilities on the right. The distance between thresholds (quantiles) determine the probability of the k outcomes.

We can also calculate and plot the predicted probabilities (i.e., $g^{-1}(\eta)$) given the predictors and the chosen regression coefficients. In this way we can try different values and see if predicted probabilities are plausible or not. The `cat_latent_plot()` and `num_latent_plot()` functions can be for respectively a categorical (Figures 7) and numerical predictor (Figures 8). In the supplementary materials we show how to use the `sim_ord_latent()` function for checking the impact of certain β for more complex models.

[Figure 7 about here]

[Figure 8 about here]

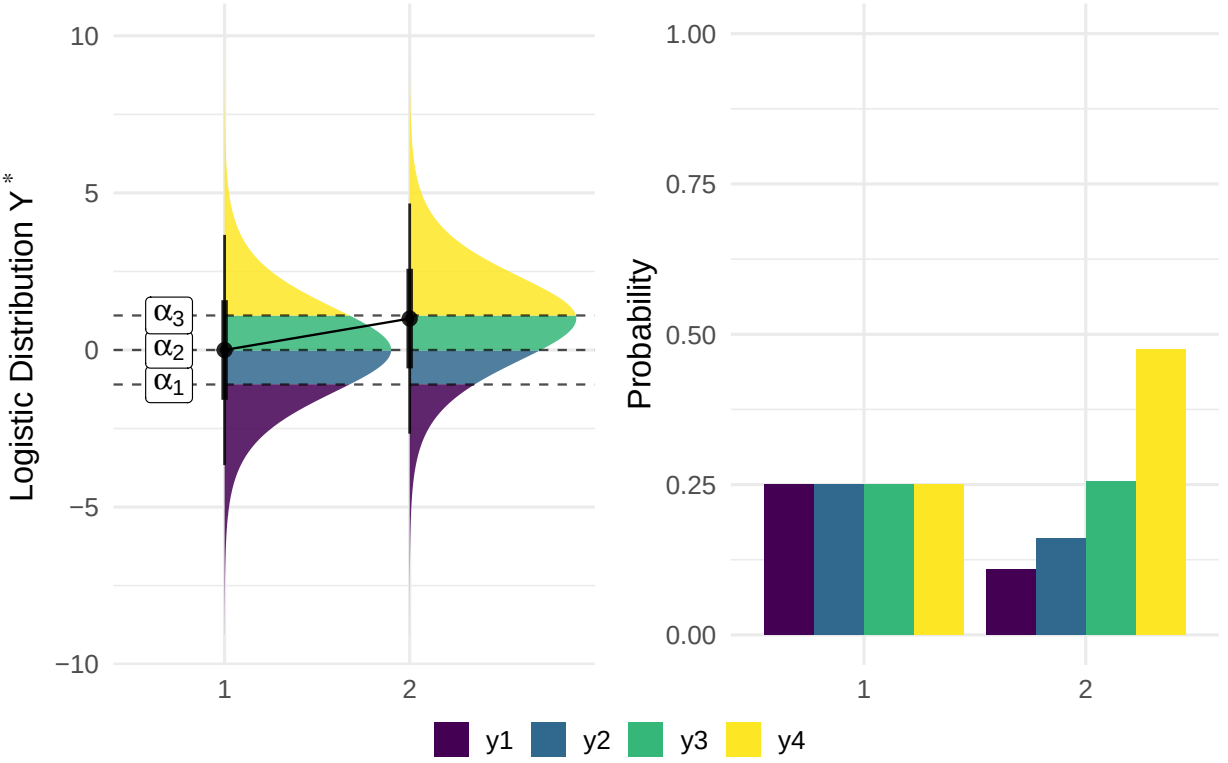

Figure 7. Example of the `cat_latent_plot()` function depicting the effect of a categorical predictor. On the left the shift in the latent mean and on the right the impact on the expected probabilities. The plot can be created using `cat_latent_plot(m = c(0, 1), s = 1, prob0 = rep(1/4, 4), link = "logit")`.

2x2 interaction

A common research design could be a 2x2 factorial design. In this example we have two main effects and the interaction. By default R use dummy coding but setting sum-to-zero contrasts (e.g., 0.5 and −0.5) for a factorial design is convenient. In this way β_1 will be the main effect of X_1, β_2 the main effect of X_2 and β_3 the interaction (thus the difference of differences). Equation (6) depict the model formula.

$$P(Y_i \leq k) = g^{-1}[\alpha_k - (\beta_1 X_{1_i} + \beta_2 X_{2_i} + \beta_3 X_{1_i} X_{2_i})] \tag{6}$$

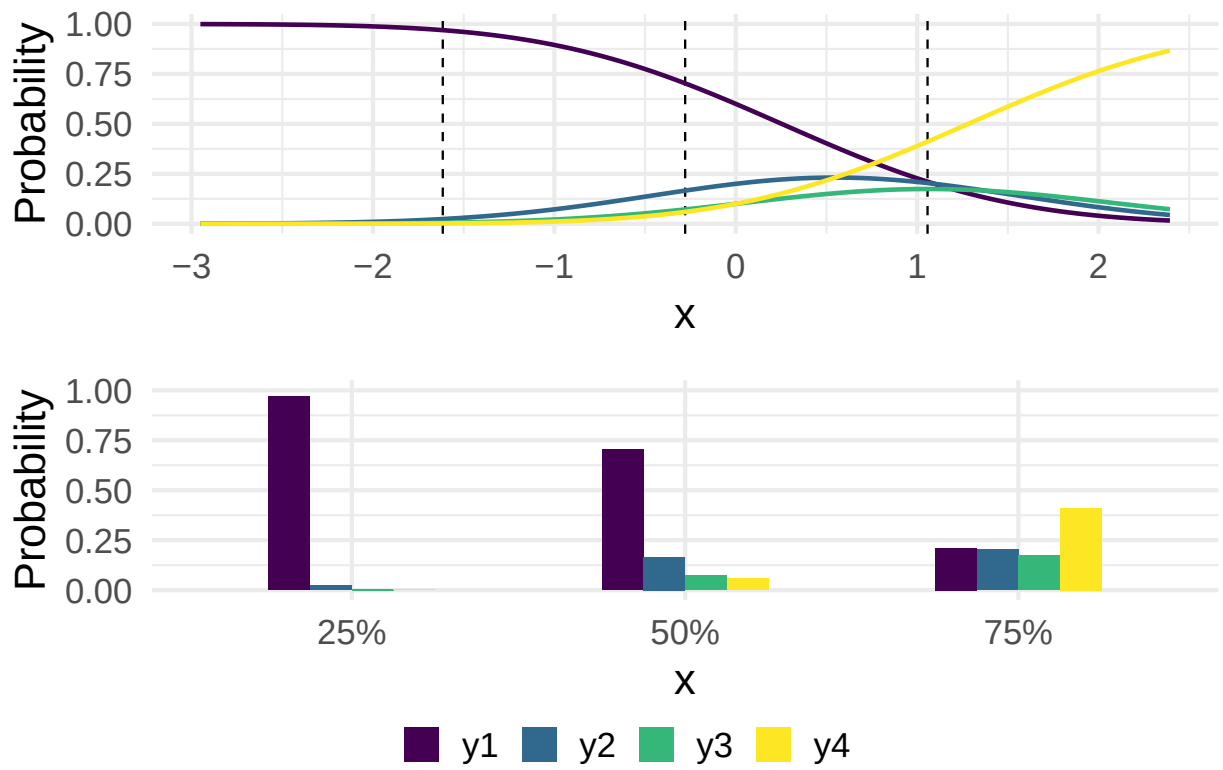

Figure 8. Example of the `num_latent_plot()` function depicting the effect of a continuous predictor on the expected probabilities. Dotted lines are the 25%, 50% and 75% quantiles. $\beta_1 = 1$ is the regression coefficient. The plot can be created using `num_latent_plot(x = runif(100), b1 = 1, prob0 = c(0.6, 0.2, 0.1, 0.1), link = "probit")`

```
set.seed(2024)
n <- 100
k <- 3
betas <- c(b1 = 0, b2 = 1, b3 = 0.5) # b1 = main effect X1, b2 = main effect X2, b3 = interaction

dat <- expand.grid(x1 = c("a", "b"), x2 = c("c", "d"), n = 1:n)
dat$x1 <- factor(dat$x1)
dat$x2 <- factor(dat$x2)

# sum to 0 coding
contrasts(dat$x1) <- c(0.5, -0.5)
contrasts(dat$x2) <- c(0.5, -0.5)
probs0 <- rep(1/k, k)
```

```r
dat <- sim_ord_latent(~ x1 * x2, beta = betas, prob0 = probs0, link = "probit", data = dat)
fit <- clm(y ~ x1 * x2, data = dat, link = "probit")
summary(fit)
#> formula: y ~ x1 * x2
#> data:    dat
#>
#>  link   threshold nobs logLik  AIC    niter max.grad cond.H
#>  probit flexible  400  -390.29 790.57 4(0)  2.20e-07 2.1e+01
#>
#> Coefficients:
#>         Estimate Std. Error z value Pr(>|z|)
#> x11     -0.02969    0.11625  -0.255    0.798
#> x21      1.15703    0.11935   9.695   <2e-16 ***
#> x11:x21  0.29058    0.23254   1.250    0.211
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Threshold coefficients:
#>     Estimate Std. Error z value
#> 1|2 -0.50503    0.06932  -7.286
#> 2|3  0.46102    0.06900   6.681
```

The thresholds are fixed, representing the probabilities $P(Y = k)$ when all predictors are zero. In this case by doing `alpha_to_prob(fit$alpha, link = "probit")` we should recover the `probs0` vector. `x11` is the main effect thus the difference in z scores between a and b averaging over `x2`. The same holds for `x21`. `x11:x21` is the interaction thus difference of differences in z scores (see Figure 9).

[Figure 9 about here]

Numerical by categorical interaction

Another common scenario is the interaction between a numerical variable x and a categorical variable g. For simplicity we simulate the factor with two levels and the numerical variable sampled from an uniform distribution between 0 and 1.

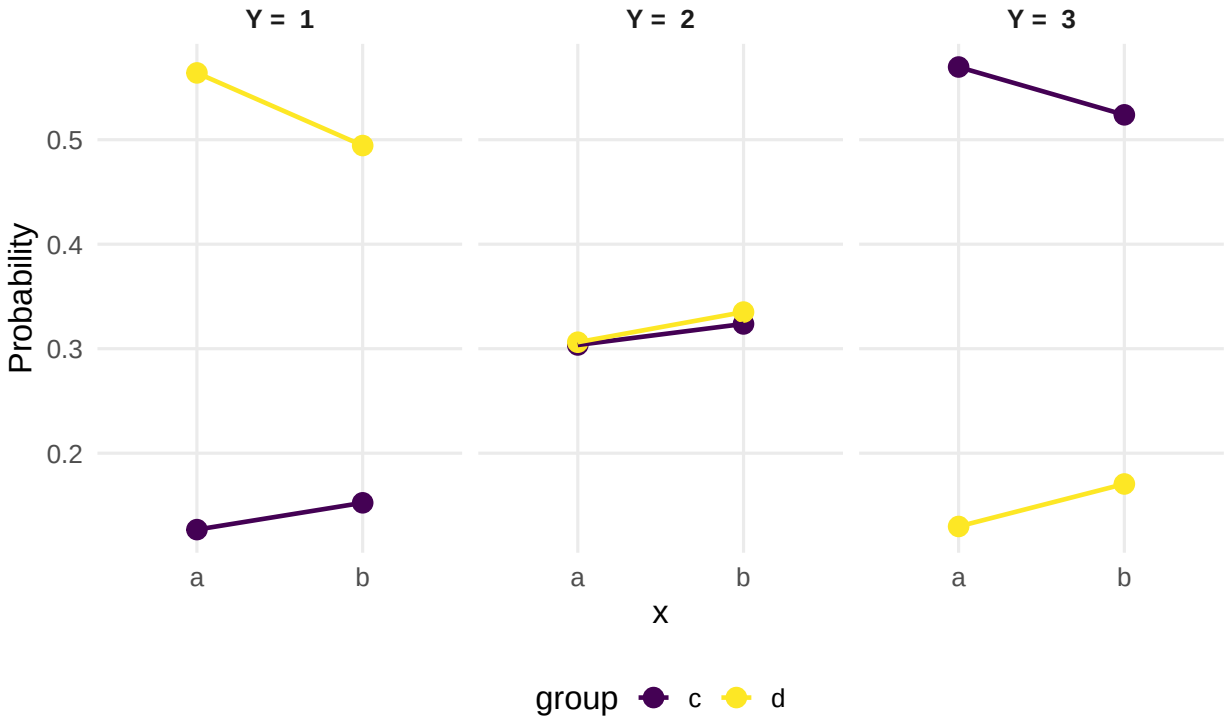

Figure 9. Results from the simulated model with a 2x2 interaction between categorical predictors. The plot shows the predicted probabilities for each ordinal outcome as a function of the two predictors. A similar plot can be produced using `plot(ggeffects::ggpredict(fit, terms = c("x1", "x2")))`.

```r
set.seed(2024)
n <- 100
k <- 3
dat <- data.frame(x = runif(n),
                  g = rep(c("a", "b"), each = n/2))
dat$g <- factor(dat$g)
contrasts(dat$g) <- c(0.5, -0.5)
probs0 <- rep(1/k, k)

dat <- sim_ord_latent(~ x * g, beta = c(0.3, 0.5, 0.5), prob0 = probs0, link = "probit", data = dat)
fit <- clm(y ~ x * g, data = dat, link = "probit")
summary(fit)
#> formula: y ~ x * g
#> data:    dat
```

```
#>
#>  link   threshold nobs logLik  AIC    niter max.grad cond.H
#>  probit flexible  100  -101.63 213.26 5(0)  3.01e-08 1.0e+02
#>
#> Coefficients:
#>      Estimate Std. Error z value Pr(>|z|)
#> x     0.68616    0.40959   1.675   0.0939 .
#> g1    0.07207    0.45768   0.157   0.8749
#> x:g1  1.21471    0.81881   1.484   0.1379
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Threshold coefficients:
#>     Estimate Std. Error z value
#> 1|2  -0.1059     0.2352  -0.450
#> 2|3   0.6529     0.2408   2.711
```

Again, `x` is the slope of the numerical predictor averaging over `g` (given that `g` has been coded with sum-to-zero contrast) thus the increase in z scores for a unit increase in `x`. `g1` is the main effect of the factor evaluated when $x = 0$ (centering x will change the parameter interpretation). The `x:g1` is the difference between the slopes of the two groups. Figure 10 depicts the model results.

[Figure 10 about here]

Power Analysis

In this section we introduce how to estimate the statistical power. Power analysis is a crucial and often necessary step when planning a confirmatory study (Lakens, 2022) or preparing a pre-registration or registered report (Chambers & Tzavella, 2022). Furthermore, statistical power for complex models can only be estimated using Monte Carlo simulations.

Formally the power is defined as the probability of correctly rejecting the null hypothesis H_0. For simple cases (e.g., a t-test) the power can be calculated analytically

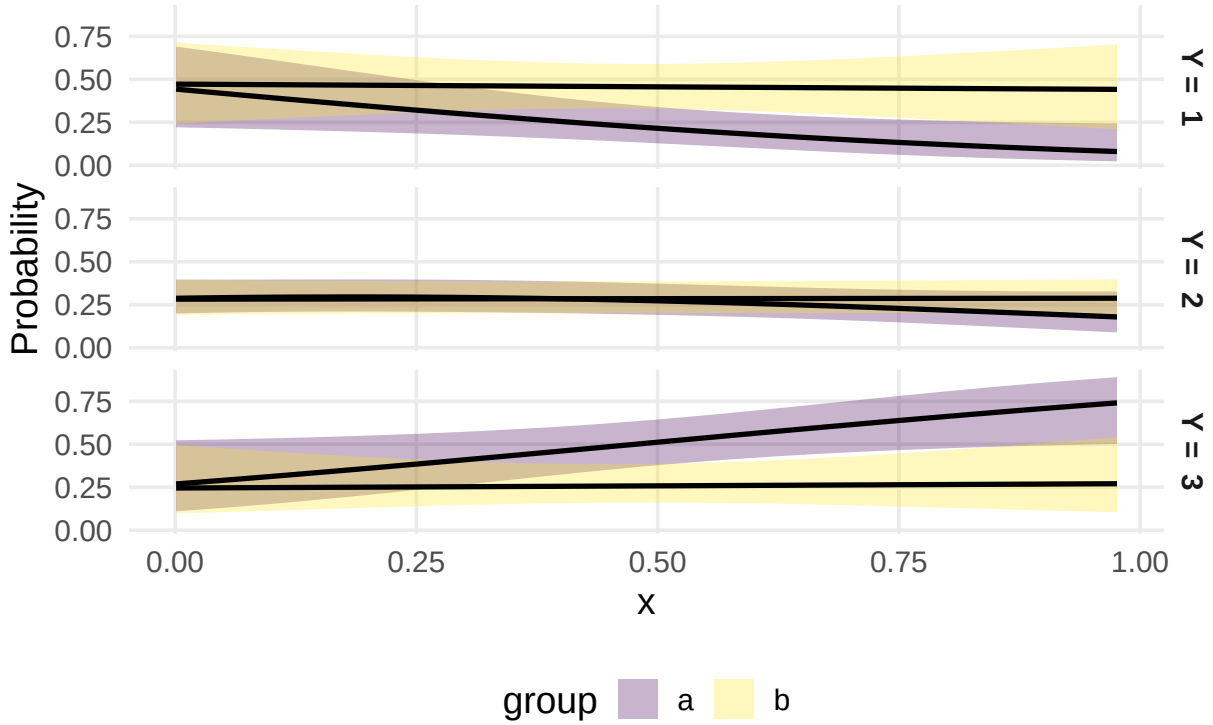

Figure 10. Results from the simulated model with the interaction between a continuous and categorical variables. The plot shows the predicted probabilities with 95% confidence intervals as a function of predictors. A similar plot can be produced using `plot(ggeffects::ggpredict(fit, terms = c("x[all]", "g")))`

while in complex cases simulating data is the best approach. We present very general example that, as for the previous simulations, can be easily extended.

The general workflow for a power simulation can be summarized as:

1. Specify the research design, e.g., 2x2 factorial design
2. Specify the effect/parameter of interest, e.g., the interaction effect
3. Define the simulation conditions, e.g., a range of sample size, effect size, etc.
4. Implement one of the simulation workflow described in the previous sections
5. Repeat the simulation a large number of times (e.g, 10000) and store the relevant values from each simulation

6. Summarise the simulation results

We can estimate the power of detecting a group difference of $d = 0.4$ (assuming a *probit* model). Participants respond to an ordinal variable with $k = 5$. The simulation is performed using a `for` loop that repeat 1000 times the data simulation, model fitting and extract the p-value for the β_1. The power is then estimated as the number of p-values lower than the critical level over the number of simulations.

```
set.seed(2024)
n <- 40 # sample size
k <- 5  # number of ordinal variables
d <- 0.4 # effect size (i.e., our regression coefficients)
nsim <- 1e3 # higher is better, here using 1000 for an example
probs0 <- rep(1/k, k)
alpha <- 0.05 # critical alpha

p <- rep(NA, nsim) # preallocation to improve loop computational efficiency
dat <- data.frame(group = rep(c("a", "b"), each = n)) # data frame

head_tail(dat, n = 3)
#>    group
#> 1      a
#> 2      a
#> 3      a
#> 78     b
#> 79     b
#> 80     b

for(i in 1:nsim){
  sim <- sim_ord_latent(~group, beta = d, prob0 = probs0, link = "probit", data = dat)
  fit <- clm(y ~ group, data = sim, link = "probit")
  p[i] <- summary(fit)$coefficients["groupb", "Pr(>|z|)"] # extract the pvalue
}

# estimate the power
mean(p <= alpha, na.rm = TRUE)
#> [1] 0.418
```

Despite useful, calculating power curves instead a single value is more informative.

For this reason we repeat the previous simulation but for different sample sizes. Figure 11 depict the power curve resulting from the simulation.

```
set.seed(2024)

n <- c(20, 40, 60, 100, 200)
power <- rep(NA, length(n))

for(i in 1:length(n)){
  p <- rep(NA, nsim) # preallocation for speed
  dat <- data.frame(group = rep(c("a", "b"), each = n[i]))
  for(j in 1:nsim){
    sim <- sim_ord_latent(~group, beta = d, prob0 = probs0, link = "probit", data = dat)
    fit <- clm(y ~ group, data = sim, link = "probit")
    p[j] <- summary(fit)$coefficients["groupb", "Pr(>|z|)"]
  }
  power[i] <- mean(p <= alpha)
}

power
#> [1] 0.225 0.412 0.522 0.750 0.964
```

[Figure 11 about here]

Conclusions

In this tutorial we presented the CM model in terms of model fitting, parameters interpretation and data simulation. The basic model can be expanded in several ways. When assuming PO or considering only a location effect is limiting, including scale (*location-scale*), thresholds (*location-shift*) effects or relaxing the PO assumption can help in modeling complex phenomenon. The supplementary materials (https://osf.io/93h5j) contains some examples of these complex models. Furthermore, with clustered data (e.g., multiple items/repetitions for the same participant), a random-effect CM model can be fitted including random intercepts and slopes.

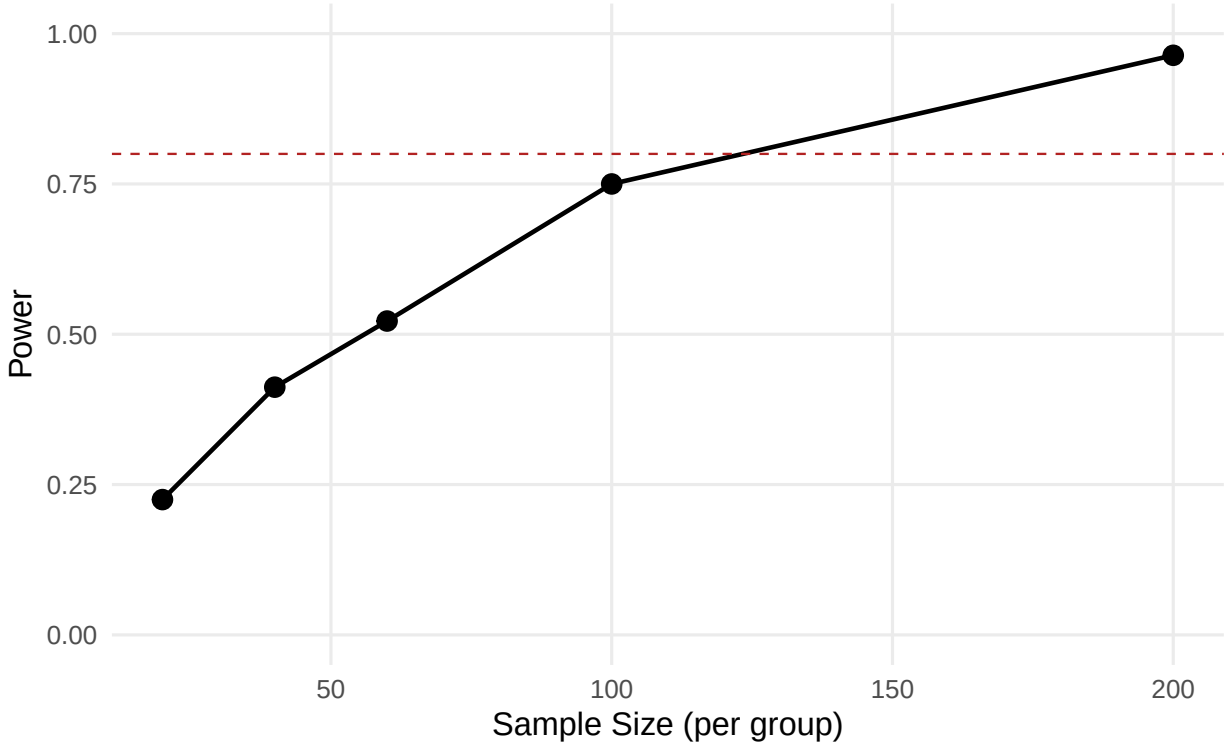

Figure 11. Results from the power analysis. The x-axis represents the sample size (per group) and the y axis the estimated power. The red dotted line highlight the 80% power level.

Ethical Compliance Section

Funding. The authors have no funding to disclose.

Compliance with Ethical Standards. Compliance with Ethical Standards: Not applicable

Conflicts of Interest. The authors declare they have no conflict of interest

References

Agresti, A. (2010). *Analysis of ordinal categorical data*. Hoboken, NJ, USA: John Wiley & Sons, Inc. https://doi.org/10.1002/9780470594001

Bürkner, P.-C., & Vuorre, M. (2019). Ordinal regression models in psychology: A tutorial. *Advances in Methods and Practices in Psychological Science*, *2*, 77–101. https://doi.org/10.1177/2515245918823199

Carifio, J., & Perla, R. (2008). Resolving the 50-year debate around using and misusing likert scales. *Medical Education*, *42*, 1150–1152. https://doi.org/10.1111/j.1365-2923.2008.03172.x

Carifio, J., & Perla, R. J. (2007). Ten common misunderstandings, misconceptions, persistent myths and urban legends about likert scales and likert response formats and their antidotes. *Journal of Social Sciences*, *3*, 106–116. https://doi.org/10.3844/jssp.2007.106.116

Chambers, C. D., & Tzavella, L. (2022). The past, present and future of registered reports. *Nature Human Behaviour*, *6*, 29–42. https://doi.org/10.1038/s41562-021-01193-7

Christensen, R. H. B. (2023). *Ordinal—regression models for ordinal data*. Retrieved from https://CRAN.R-project.org/package=ordinal

Cliff, N. (1996). Answering ordinal questions with ordinal data using ordinal statistics. *Multivariate Behavioral Research*, *31*, 331–350. https://doi.org/10.1207/s15327906mbr3103_4

Cox, C. (1995). Location-scale cumulative odds models for ordinal data: A generalized non-linear model approach. *Statistics in Medicine*, *14*, 1191–1203. https://doi.org/10.1002/sim.4780141105

DeBruine, L. M., & Barr, D. J. (2021). Understanding mixed-effects models through data simulation. *Advances in Methods and Practices in Psychological Science*, *4*, 2515245920965119. https://doi.org/10.1177/2515245920965119

DeCarlo, L. T. (2010). On the statistical and theoretical basis of signal detection theory

and extensions: Unequal variance, random coefficient, and mixture models. *Journal of Mathematical Psychology*, *54*, 304–313. https://doi.org/10.1016/j.jmp.2010.01.001

Fox, J. (2015). *Applied regression analysis and generalized linear models*. SAGE Publications.

Gambarota, F., & Altoè, G. (2023). Understanding meta-analysis through data simulation with applications to power analysis. *Advances in Methods and Practices in Psychological Science*. https://doi.org/10.1177/25152459231209330

Gelman, A., Hill, J., & Vehtari, A. (2020). *Regression and other stories*. Cambridge University Press. https://doi.org/10.1017/9781139161879

Jamieson, S. (2004). Likert scales: How to (ab)use them. *Medical Education*, *38*, 1217–1218. https://doi.org/10.1111/j.1365-2929.2004.02012.x

Kemp, S., & Grace, R. C. (2021). Using ordinal scales in psychology. *Methods in Psychology (Online)*, *5*, 100054. https://doi.org/10.1016/j.metip.2021.100054

Lakens, D. (2022). Sample size justification. *Collabra. Psychology*, *8*. https://doi.org/10.1525/collabra.33267

Liddell, T. M., & Kruschke, J. K. (2018). Analyzing ordinal data with metric models: What could possibly go wrong? *Journal of Experimental Social Psychology*, *79*, 328–348. https://doi.org/10.1016/j.jesp.2018.08.009

Likert, R. (1932). A technique for the measurement of attitudes. *Archives of Psychology*, *22*, 55.

Liu, A., He, H., Tu, X. M., & Tang, W. (2023). On testing proportional odds assumptions for proportional odds models. *General Psychiatry*, *36*, e101048. https://doi.org/10.1136/gpsych-2023-101048

McCullagh, P. (1980). Regression models for ordinal data. *Journal of the Royal Statistical Society*, *42*, 109–127. https://doi.org/10.1111/j.2517-6161.1980.tb01109.x

Overgaard, M., & Sandberg, K. (2021). The perceptual awareness scale—recent controversies and debates. *Neuroscience of Consciousness*.

https://doi.org/10.1093/nc/niab044/41765024/niab044

Peterson, B., & Harrell, F. E. (1990). Partial proportional odds models for ordinal response variables. *Journal of the Royal Statistical Society. Series C, Applied Statistics*, *39*, 205. https://doi.org/10.2307/2347760

R Core Team. (2023). *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing. Retrieved from https://www.R-project.org/

Rigby, R. A., & Stasinopoulos, D. M. (2005). Generalized additive models for location, scale and shape. *Journal of the Royal Statistical Society. Series C, Applied Statistics*, *54*, 507–554. https://doi.org/10.1111/j.1467-9876.2005.00510.x

Robitzsch, A. (2020). Why ordinal variables can (almost) always be treated as continuous variables: Clarifying assumptions of robust continuous and ordinal factor analysis estimation methods. *Frontiers in Education*, *5*, 589965. https://doi.org/10.3389/feduc.2020.589965

Sánchez-Meca, J., Marín-Martínez, F., & Chacón-Moscoso, S. (2003). Effect-size indices for dichotomized outcomes in meta-analysis. *Psychological Methods*, *8*, 448–467. https://doi.org/10.1037/1082-989X.8.4.448

Schad, D. J., Vasishth, S., Hohenstein, S., & Kliegl, R. (2020). How to capitalize on a priori contrasts in linear (mixed) models: A tutorial. *Journal of Memory and Language*, *110*, 104038. https://doi.org/10.1016/j.jml.2019.104038

Stevens, S. S. (1946). On the theory of scales of measurement. *Science (New York, N.Y.)*, *103*, 677–680. https://doi.org/10.1126/science.103.2684.677

Tutz, G. (1990). Sequential item response models with an ordered response. *The British Journal of Mathematical and Statistical Psychology*, *43*, 39–55. https://doi.org/10.1111/j.2044-8317.1990.tb00925.x

Tutz, G. (2022). Ordinal regression: A review and a taxonomy of models. *Wiley Interdisciplinary Reviews. Computational Statistics*, *14*.

https://doi.org/10.1002/wics.1545

Tutz, G., & Berger, M. (2020). Non proportional odds models are widely dispensable – sparser modeling based on parametric and additive location-shift approaches. *arXiv [Stat.ME]*. Retrieved from http://arxiv.org/abs/2006.03914