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

Supplementary Materials

Filippo Gambarota Gianmarco Altoè

## Contents

```r
xl <- seq(0, 1, 0.1)
x <- rep(xl, 10)
n <- length(x)
b0 <- qlogis(0.01)
b1 <- 8
lp <- b0 + b1 * x # linear predictor
y <- rbinom(n, 1, plogis(lp)) # sampling from a binomial distribution using the vector of probabilities
p <- tapply(y, x, mean)
plot(unique(x), p, pch = 19)
curve(plogis(x, -(b0/b1), 1/b1), add = TRUE, lwd = 2, col = "firebrick")
```

Essentially the `lp` ($\eta_i$) vector contains true probability of success ($p_i|x_i = g^{-1}(\eta_i)$) for the $x_i$ level of the predictor. Then to introduce the random part of the model, we can use $p_i$ to generate binary values from a Binomial (in this specific case a Bernoulli) distribution. This create `y` that is the vector of 0-1 values generated using the true probabilities of success. The supplementary materials contains a more extensive example for simulating data for generalized linear models.

We can apply the same idea to an ordinal outcome but we need $k-1$ equations (compared to the previous example) where $k$ is the number of ordinal levels. Let's simulate a similar design with a continuous $x$ predictor and $k = 4$ options. We fix the baseline probabilities where $x = 0$ as uniform thus $p(y_1) = p(y_2) = ...p(y_k) = 1/k$.

```r
k <- 4  # number of options (1)
n <- 1e5 # number of observations (1)
b1 <- 0.5 # beta1, the shift in the latent distribution (2)
probs0 <- rep(1/k, k) # probabilities when x = 0 (3)
alphas <- prob_to_alpha(probs0, link = "probit") # get true thresholds from probabilities (3)
dat <- data.frame(x = rnorm(n)) # create dataframe
lp <- lapply(alphas, function(a) a - b1 * dat$x) # k - 1 linear predictors (4)
names(lp) <- sprintf("g[p(y <= %s)]", 1:(k - 1)) # giving appropriate names
```

The linear predictors ($\eta$) can be seen in the Figure ref. The $g$ in this case is the cumulative *probit* function $\Phi$.

Now we can apply the inverse of the link function $g^{-1} = \Phi^{-1}$ to calculate the corresponding cumulative probabilities.

```r
cump <- lapply(lp, pnorm) # inverse of the link function (5)
cump <- data.frame(cump)
p <- apply(cbind(0, cump, 1), 1, diff, simplify = FALSE) # probability of each k outcome (6)
p <- do.call(rbind, p)
```

```r
p <- data.frame(p)
names(p) <- sprintf("p_y%s", 1:k)
filor::trim_df(data.frame(p))
```

We can plot the expected effect of $\beta_1$ on the $k$ probabilities using the `num_latent_plot()` function (see Figure **??**). Finally we sample using the `sample()` function using the calculated probabilities.

```r
num_latent_plot(x = dat$x, b1 = b1, probs = probs0, link = "probit")
```

```r
dat$y <- apply(p, 1, function(ps) sample(1:k, 1, prob = ps))
```

## 0.1 Scale Effects

The default ordinal regression model assume that the variance of the underlying latent distribution is the same across condition. This is similar to a standard linear regression assuming the homogeneity of variance. For example, when comparing two groups or conditions we can run a standard linear model (i.e., a t-test) assuming homogeneity of variances or using the Welch t-test [see @Delacre2017-qy]. In addition, there are the so-called location-scale models that allows to include predictors also for the scale (e.g., the variance) of the distribution. This can be done also in ordinal regression where instead of assuming the same variance between conditions, the linear predictors can be included. The Equation (number here) expand the previous model including the linear predictor on the scale of the latent distribution.

$$g(P(Y \leq k)) = \frac{\alpha_k - X\beta}{e^{X\zeta}} \tag{S1}$$

$$Y_i^\star = \eta + \epsilon_i \epsilon_i \sim \mathcal{N}(0, e^{X\zeta}) \tag{S2}$$

Where $X\zeta$ is the linear predictor $\eta$ for the scale of the distribution. By dafult for both the logit and probit model the scale is fixed to 1. On scale-location models we put predictors on both parameters. Given that the scale cannot be negative we use a log link function $\eta = \log(X\zeta)$. Figure **??** depict an example of a comparison between two groups where the two underlying latent distributions have unequal variance. Ignoring the heterogeneity of variances could lead to underestimation of the effect size, reduced power and inflated type-1 error rate (qualche ref qui). Furtermore, as suggested by @Tutz2017-de location-scale models can be considered as a more parsimonius approach compared to partially or completely relaxing the proportional odds assumption. Allowing the scale to be different as a function of the predictor create more modelling flexbility. Furthemore, two groups could be theoretically different only in the scale of the latent distribution with a similar location. In this example, the only way to capture group differences is by including a scale effect. The Figure **??** depict the impact of having different scales between two groups on the ordinal probabilities.

@Tutz2022-dg provide a very clear and intuitive explanation of what happen when including a scale effect and how to interpret the result. As suggested before, the scale-location model allow to independently predict changes in the location and the scale. While location shifts are simply interpreted as increasing/decreasing the latent $\mu$ or the odds of responding a certain category scale effects are not straightforward. As the scale increase (e.g., the variance increase) there is an higher probability mass on extreme categories. On the other side as the scale decrease, responses are more concentrated on single categories. The categories are

determined by the location parameter. For example, if one group have a certain latent mean $\mu_1$ and a small scale $\sigma^2 = 1/3$ (thus one third compared to the standard version of the distribution), all responses will be focused on categories around the latent mean. On the other side, increasing the scale will increase the cumulative probabilities for all categories and for values that tends to infinity extreme categories are preferred. Clearly, the scale parameter can somehow be interpreted as the response style (concentrated or variable). This is conceptually similar but implemented and interpreted differently to shift-location models [@Tutz2022-dg; Tutz2020-xq] where thresholds are allowed to vary as a function of predictors. Both models tries to increase the flexibility of modelling the probability structure of the responses by adding extra paprameters to predict the probabilities of the ordinal responses.

```
sigma_high <- cat_latent_plot(m = c(0, 0),
                s = c(1, 2),
                probs = rep(1/4, 4),
                link = "probit",
                plot = "both",
                title = latex2exp::TeX("$\\sigma^2_{g_2} = 2$"))
sigma_low <- cat_latent_plot(m = c(0, 0),
                s = c(1, 0.5),
                probs = rep(1/4, 4),
                link = "probit",
                plot = "both",
                title = latex2exp::TeX("$\\sigma^2_{g_2} = 0.5$"))
sigma_high_mu1 <- cat_latent_plot(m = c(0, 1),
                s = c(1, 2),
                probs = rep(1/4, 4),
                link = "probit",
                plot = "both",
                title = latex2exp::TeX("$\\sigma^2_{g_2} = 2$, $\\mu_2 = 1$"))
sigma_low_mu1 <- cat_latent_plot(m = c(0, 1),
                s = c(1, 0.5),
                probs = rep(1/4, 4),
                link = "probit",
                plot = "both",
                title = latex2exp::TeX("$\\sigma^2_{g_2} = 0.5$, $\\mu_2 = 1$"))
plot_grid(sigma_high, sigma_low, sigma_high_mu1, sigma_low_mu1)
```

### 0.1.1 Simulating scale effects

The location-scale model can be simulated using the `sim_ord_latent()` function and providing the predictors for the scale parameters. Given the `log` link function, predictors are provided on the `log` scale. For example, we simulate the effect of a binary variable $x$ representing two independent groups predicting the $k = 5$ response. We simulate a *location* effect of $\beta_1 = 0.5$ (in *probit* scale) and $\zeta_1 = \log(2) = 0.70$. The first group has a $\sigma = 1$ and the second group has $\sigma = 2$. Again we simulate that the baseline probabilities are uniform for the first group.

```
k <- 5  # number of options (1)
n <- 1e5 # number of observations (1)
b1 <- 0.5 # beta1, the shift in the latent distribution (2)
z1 <- log(2) # zeta1, the change in the scale
probs0 <- rep(1/k, k) # probabilities when x = 0 (3)
alphas <- prob_to_alpha(probs0, link = "probit") # get true thresholds from probabilities (3)
dat <- data.frame(x = rep(c(0, 1), each = n/2))
```

```r
dat <- sim_ord_latent(~x, scale = ~x, By = b1, Bscale = z1, prob = probs0, data = dat, link = "probit")
fit <- clm(y ~ x, scale = ~ x, data = dat, link = "probit")
```

The table (put ref) reports the simulation results.

```r
truth <- c(alphas, b1, z1)
names(truth) <- c(names(fit$alpha), names(fit$beta), names(fit$zeta))
#clm_table(fit, truth)
```

To better understand the impact of assuming (or simulating) a different latent scale we fit $k-1$ binomial regressions and check the estimated coefficients. We are not simulating a specific beta for each outcome but simulating a scale effect is actually impacting the regression coefficients. When generating data for a binary outcome the linear predictor is composed by $\eta = \beta_0 + \beta_1 x$. The threshold $\alpha$ and slope of the function can be estimated using $\alpha = -\frac{\beta_0}{\beta_1}$ and the slope is $\frac{1}{\beta_1}$ [@Knoblauch2012-to; @Faraggi2003-hm]. Under the proportional odds assumption, there is only a change in thresholds $\alpha$ this a shift in the sigmoid along the $x$ axis. When including a scale effect a change in the sigmoid is combined with a change in the slope.

```r
set.seed(2023)
k <- 4
n <- 1e5
b1 <- 3
d1 <- log(2)
dat <- data.frame(x = runif(n))
dat <- sim_ord_latent(~x, ~x, By = b1, Bscale = d1, prob = rep(1/k, k), data = dat, link = "probit")

dat$y1vs234 <- ifelse(dat$y <= 1, 1, 0)
dat$y12vs34 <- ifelse(dat$y <= 2, 1, 0)
dat$y123vs4 <- ifelse(dat$y <= 3, 1, 0)

dat$y <- ordered(dat$y)
fit <- clm(y ~ x, scale = ~x, data = dat, link = "probit")

fit1vs234 <- glm(y1vs234 ~ x, data = dat, family = binomial(link = "probit"))
fit12vs34 <- glm(y12vs34 ~ x, data = dat, family = binomial(link = "probit"))
fit123vs4 <- glm(y123vs4 ~ x, data = dat, family = binomial(link = "probit"))

fits <- list(y1vs234 = fit1vs234, fit12vs34 = fit12vs34, fit123vs4 = fit123vs4)

lapply(fits, function(x) coef(x)) |>
  bind_rows(.id = "model") |>
  mutate(xp = list(seq(0, 5, 0.01))) |>
  unnest(xp) |>
  ggplot(aes(x = xp, y = `(Intercept)` + x * xp)) +
  geom_line(aes(color = model))
```