Multinomial Regression and Interpretation: It depends on 'where we start'!

Version 0.1

Holger Sennhenn-Reulen[•]

[•]Northwest German Forest Research Institite (NW-FVA)

April 6, 2022

Abstract Paolino (2020) writes with respect to the interpretability of multinomial regression models: 'A focus upon the statistical significance of predicted probabilities and marginal effects provides several benefits.' I very much agree with this point, in especially in a Bayesian framework with posterior (Monte Carlo) samples that make calculating these probabilities – with their uncertainties – pretty straightforward! But: one needs to be cautios of the 'effects' of non-linear transformations then. That is what this short note is about!

Contents

1 Multinomial Regression

The multinomial logit link function[1] is defined as:

$$\Pr\left(Y_i = k\right) = \frac{\exp\left(\eta_{k,i}\right)}{\sum_{\tilde{k}=1}^{K} \exp\left(\eta_{\tilde{k},i}\right)}$$

Here one needs to set one of the response categories as reference, in `brms` (Bürkner, 2017, 2018) this is the first category. For this reference category, the linear predictor is equal to a value of 0 for all observation units i.
Let's build an example in which we have a categorical outcome Y with three categories $\{A, B, C\}$. Category A is our reference. We further take into consideration two covariates, x_1 and x_2, of which each has some 'true effect', $\beta_{1,B}$, $\beta_{1,C}$, $\beta_{2,B}$, and $\beta_{2,C}$, in both linear predictors:

$$\exp\left(\beta_{0,B} + x_{1,i}\beta_{1,B} + x_{2,i}\beta_{2,B}\right) = \frac{\Pr\left(Y_i = B \mid x_{1,i} = 1, x_{2,i}\right)}{\Pr\left(Y_i = A \mid x_{1,i} = 1, x_{2,i}\right)},$$
$$\exp\left(\beta_{0,C} + x_{1,i}\beta_{1,C} + x_{2,i}\beta_{2,C}\right) = \frac{\Pr\left(Y_i = C \mid x_{1,i} = 1, x_{2,i}\right)}{\Pr\left(Y_i = A \mid x_{1,i} = 1, x_{2,i}\right)},$$

Now expanding the left-hand side of each of the equations[2]:

$$\exp\left(\beta_{0,B} + x_{1,i}\beta_{1,B} + x_{2,i}\beta_{2,B}\right) = \exp\left(\beta_{0,B}\right)\exp\left(x_{1,i}\beta_{1,B}\right)\exp\left(x_{2,i}\beta_{2,B}\right),$$
$$\exp\left(\beta_{0,C} + x_{1,i}\beta_{1,C} + x_{2,i}\beta_{2,C}\right) = \exp\left(\beta_{0,C}\right)\exp\left(x_{1,i}\beta_{1,C}\right)\exp\left(x_{2,i}\beta_{2,C}\right),$$

gives us the 'usual' interpretation of the regression parameters as a covariate weight for a multiplicative change in the probability – conditional on $x_1 = x_{1,i}$ and $x_2 = x_{2,i}$ – for observing category k in comparison to the probability – again, conditional on $x_1 = x_{1,i}$ and $x_2 = x_{2,i}$ – for observing the reference category:

$$\exp\left(\beta_{0,B}\right)\exp\left(x_{1,i}\beta_{1,B}\right)\exp\left(x_{2,i}\beta_{2,B}\right) = \frac{\Pr\left(Y_i = B \mid x_{1,i} = 1, x_{2,i}\right)}{\Pr\left(Y_i = A \mid x_{1,i} = 1, x_{2,i}\right)},$$
$$\exp\left(\beta_{0,C}\right)\exp\left(x_{1,i}\beta_{1,C}\right)\exp\left(x_{2,i}\beta_{2,C}\right) = \frac{\Pr\left(Y_i = C \mid x_{1,i} = 1, x_{2,i}\right)}{\Pr\left(Y_i = A \mid x_{1,i} = 1, x_{2,i}\right)}.$$

Let's make a first example for our inputs and set $x_{1,i} = 0$:

$$\exp\left(\beta_{0,B} + x_{2,i}\beta_{2,B}\right) = \frac{\Pr\left(Y_i = B \mid x_{1,i} = 0, x_{2,i}\right)}{\Pr\left(Y_i = A \mid x_{1,i} = 0, x_{2,i}\right)}$$

And now let's set $x_{1,i} = 1$:

$$\exp\left(\beta_{0,B} + \beta_{1,B} + x_{2,i}\beta_{2,B}\right) = \frac{\Pr\left(Y_i = B \mid x_{1,i} = 1, x_{2,i}\right)}{\Pr\left(Y_i = A \mid x_{1,i} = 1, x_{2,i}\right)}$$

We can write the left-hand side as:

$$\exp\left(\beta_{0,B} + \beta_{1,B} + x_{2,i}\beta_{2,B}\right) = \exp\left(\beta_{1,B}\right)\exp\left(\beta_{0,B} + x_{2,i}\beta_{2,B}\right)$$

and consequently plug in the $x_{1,i} = 0$, and solve with respect to $\exp\left(\beta_{1,B}\right)$:

$$\exp\left(\beta_{1,B}\right) = \frac{\left(\frac{\Pr\left(Y_i = B \mid x_{1,i} = 1, x_{2,i}\right)}{\Pr\left(Y_i = A \mid x_{1,i} = 1, x_{2,i}\right)}\right)}{\left(\frac{\Pr\left(Y_i = B \mid x_{1,i} = 0, x_{2,i}\right)}{\Pr\left(Y_i = A \mid x_{1,i} = 0, x_{2,i}\right)}\right)}$$

So this interpretation is, as Paolino (2020) argues, often misses answering the applied research question.
Therefore, one should instead calculate the predicted probabilities, and base statements interpretation of $\beta_{j,k}$ on these. So, Paolino (2020) give differences of calculated predicted probabilities (Table 1) as the basis for investigating influences of the input variables. However, as Paolino (2020) further argues:

> "a critical aspect is that a covariate's effect upon the probability of observing each outcome is a function of all of the estimated coefficients."

This will be investigated further in the following.

[1] In the Stan (Stan Development Team, 2021) community also called *softmax* function.
[2] Using $\exp\left(a + b\right) = \exp\left(a\right)\exp\left(b\right)$.

2 Organize R Session

I use R (R Core Team, 2020)! Modeling is performed using `brms` (Bürkner, 2017, 2018), which is based on probabilistic programming language Stan (Stan Development Team, 2021). Graphics are generated using `ggplot2` (Wickham, 2016), `ggdist` (Kay, 2021), and `beeswarm` (Eklund and Trimble, 2021).
Some summaries are calculated using `plyr` (Wickham, 2011).
Load used packages:

```
> library("ggplot2")
> library("ggdist")
> library("brms")
> library("plyr")
```

Session info:

```
> sessionInfo()
```

```
R version 3.6.3 (2020-02-29)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 18.04.6 LTS

Matrix products: default
BLAS:   /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.7.1
LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.7.1

locale:
 [1] LC_CTYPE=de_DE.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=de_DE.UTF-8        LC_COLLATE=de_DE.UTF-8
 [5] LC_MONETARY=de_DE.UTF-8    LC_MESSAGES=de_DE.UTF-8
 [7] LC_PAPER=de_DE.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=de_DE.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] plyr_1.8.4    brms_2.16.0   Rcpp_1.0.6    ggdist_3.0.0  ggplot2_3.3.5

loaded via a namespace (and not attached):
 [1] nlme_3.1-157         matrixStats_0.61.0   xts_0.11-2
 [4] threejs_0.3.1        rstan_2.21.3         tensorA_0.36.1
 [7] tools_3.6.3          backports_1.2.1      R6_2.5.0
[10] DT_0.9               mgcv_1.8-40          projpred_2.0.2
[13] colorspace_2.0-1     withr_2.4.2          tidyselect_1.1.0
[16] gridExtra_2.3        prettyunits_1.1.1    processx_3.4.1
[19] Brobdingnag_1.2-6    emmeans_1.6.2-1      compiler_3.6.3
[22] cli_2.5.0            shinyjs_1.0          sandwich_2.5-1
[25] colourpicker_1.0     posterior_1.0.1      scales_1.0.0
[28] dygraphs_1.1.1.6     checkmate_1.9.4      mvtnorm_1.1-2
[31] ggridges_0.5.1       callr_3.3.2          stringr_1.4.0
[34] digest_0.6.27        StanHeaders_2.21.0-7 minqa_1.2.4
[37] base64enc_0.1-3      pkgconfig_2.0.3      htmltools_0.5.1.1
[40] lme4_1.1-21          htmlwidgets_1.3      rlang_0.4.11
[43] shiny_1.3.2          farver_2.1.0         generics_0.1.0
[46] zoo_1.8-9            crosstalk_1.0.0      gtools_3.9.2
[49] dplyr_1.0.2          distributional_0.2.2 inline_0.3.19
```

```
[52] magrittr_2.0.3       loo_2.3.1            bayesplot_1.7.0
[55] Matrix_1.4-1         munsell_0.5.0        abind_1.4-5
[58] lifecycle_0.2.0      multcomp_1.4-11      stringi_1.6.2
[61] MASS_7.3-56          pkgbuild_1.3.1       grid_3.6.3
[64] parallel_3.6.3       promises_1.0.1       crayon_1.4.1
[67] miniUI_0.1.1.1       lattice_0.20-45      splines_3.6.3
[70] ps_1.6.0             pillar_1.4.6         igraph_1.2.6
[73] boot_1.3-28          estimability_1.3     markdown_1.1
[76] shinystan_2.5.0      codetools_0.2-18     reshape2_1.4.3
[79] stats4_3.6.3         rstantools_2.1.1     glue_1.4.2
[82] RcppParallel_5.1.4   vctrs_0.3.4          nloptr_1.2.2.2
[85] httpuv_1.5.2         gtable_0.3.0         purrr_0.3.4
[88] mime_0.10            xtable_1.8-4         coda_0.19-4
[91] later_0.8.0          survival_3.3-1       rsconnect_0.8.15
[94] tibble_3.0.4         shinythemes_1.1.2    gamm4_0.2-6
[97] TH.data_1.0-10       ellipsis_0.3.0       bridgesampling_0.7-2
```

3 An artificial truth

We set up a simple framework with a categorial outcome with three categories and two input variables, x_1 and x_2. We set the first outcome category as reference, set the parameter for x_1 on the linear predictor for the second category on 1, that for x_2 on 0, and for the third category, we set the parameter for x_1 on 0, and that for x_2 on 1.
Given those true values …

```
> get_P <- function(x1, x2) {
+   eta1 <- 1 * x1 + 0 * x2
+   eta2 <- 0 * x1 + 1 * x2
+   tmp <- exp(0) + exp(eta1) + exp(eta2)
+   P <- cbind(exp(0)/tmp, exp(eta1)/tmp, exp(eta2)/tmp)
+   return(P)
+ }
```

… we can calculate the conditional probabilities for the three outcome categories:

```
> round(get_P(0, 0), 3)

      [,1]  [,2]  [,3]
[1,] 0.333 0.333 0.333

> round(get_P(0, 1), 3)

      [,1]  [,2]  [,3]
[1,] 0.212 0.212 0.576

> round(get_P(1, 0), 3)

      [,1]  [,2]  [,3]
[1,] 0.212 0.576 0.212

> round(get_P(1, 1), 3)

      [,1]  [,2]  [,3]
[1,] 0.155 0.422 0.422
```

What are the differences?

```
> round(get_P(0, 0) - get_P(0, 1), 3)

      [,1]  [,2]   [,3]
[1,] 0.121 0.121 -0.243

> round(get_P(1, 0) - get_P(1, 1), 3)

      [,1]  [,2]  [,3]
[1,] 0.057 0.154 -0.21
```

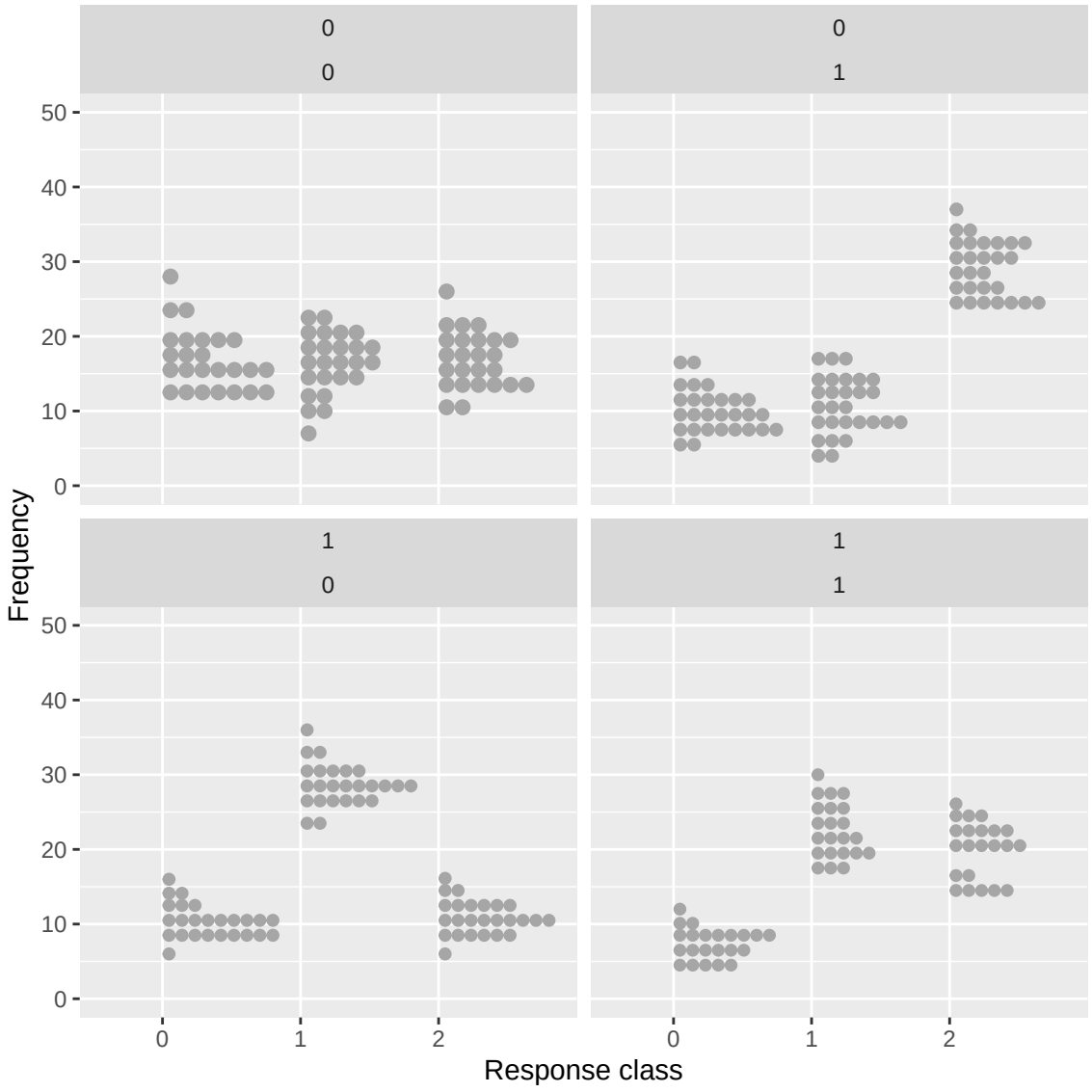
… this is what Paolino (2020) means when writing that the predicted probabilities depend on every parameter of the model: It is not 'while holding all else equal' / 'ceteris paribus' anymore on this scale!
The same does not only hold for differences in the predicted probabilities, but also for ratios:

```
> round(get_P(0, 0) / get_P(0, 1), 3)

      [,1]  [,2]  [,3]
[1,] 1.573 1.573 0.579

> round(get_P(1, 0) / get_P(1, 1), 3)

      [,1]  [,2]  [,3]
[1,] 1.364 1.364 0.502
```

4 Simulation

We simulate 100 observation units, and for each, generate a sample of size 50 from the multinomial distribution.

```
> N <- 100
> set.seed(123)
> df <- data.frame(x1 = rbinom(n = N, size = 1, prob = .5),
+                  x2 = rbinom(n = N, size = 1, prob = .5),
+                  y0 = NA,
+                  y1 = NA,
+                  y2 = NA)
> eta1 <- 1 * df$x1 + 0 * df$x2
> eta2 <- 0 * df$x1 + 1 * df$x2
> tmp <- exp(0) + exp(eta1) + exp(eta2)
> P <- cbind(exp(0)/tmp, exp(eta1)/tmp, exp(eta2)/tmp)
> rm(tmp)
> size <- 50
> for (i in 1:N) {
+   df[i, c("y0", "y1", "y2")] <- rmultinom(n = 1, size = size,
+                                          prob = P[i, ])[, 1]
+ }
```



5 Model

When using `brms`, we can provide the `size` using the additional response attribute `trials`:

```
> df$size <- size
> df$y <- cbind(df$y0, df$y1, df$y2)

> frmla <- bf(y | trials(size) ~ x1 + x2)
> pr <- get_prior(formula = frmla, family = "multinomial", data = df)
> pr$prior[which(pr$coef != "" & pr$class == "b")] <- "normal(0, 2.5)"
> m <- brm(formula = frmla, family = "multinomial", prior = pr, data = df,
+          chains = 4, cores = 4, iter = 2000, seed = 123456789)
```

6 Results

```
> summary(m, prior = T)

 Family: multinomial
  Links: mu2 = logit; mu3 = logit
Formula: y | trials(size) ~ x1 + x2
   Data: df (Number of observations: 100)
  Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup draws = 4000

Priors:
b_mu2_x1 ~ normal(0, 2.5)
b_mu2_x2 ~ normal(0, 2.5)
b_mu3_x1 ~ normal(0, 2.5)
b_mu3_x2 ~ normal(0, 2.5)
Intercept_mu2 ~ student_t(3, 0, 2.5)
Intercept_mu3 ~ student_t(3, 0, 2.5)

Population-Level Effects:
             Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
mu2_Intercept    -0.03      0.06    -0.15     0.09 1.00     3985     2973
mu3_Intercept     0.03      0.06    -0.10     0.15 1.00     4068     3068
mu2_x1            1.05      0.08     0.89     1.19 1.00     3356     2804
mu2_x2            0.12      0.08    -0.03     0.27 1.00     3164     2790
mu3_x1           -0.01      0.08    -0.16     0.15 1.00     3134     2351
mu3_x2            1.03      0.08     0.88     1.18 1.00     3221     2870

Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).

> nd <- data.frame(size = 1,
+                  x1 = c(0, 0, 1, 1),
+                  x2 = c(0, 1, 0, 1))
> fit <- fitted(m, newdata = nd, summary = F, scale = "linear")
> dim(fit)

[1] 4000    4    2

> tmp <- 1 + apply(exp(fit), MAR = c(1, 2), FUN = sum)
> ## quick check:
> head(tmp)

         [,1]     [,2]     [,3]     [,4]
[1,] 3.151088 4.650438 5.117128 6.457291
[2,] 2.990901 4.749001 4.770535 6.759048
[3,] 2.987780 4.876176 4.960417 7.068784
[4,] 2.857211 4.815935 4.522580 6.770756
[5,] 2.868259 4.887150 4.577038 6.860803
[6,] 3.088579 5.274816 4.651858 6.959012

> 1 + exp(fit[1:6, , 1]) + exp(fit[1:6, , 2])

         [,1]     [,2]     [,3]     [,4]
[1,] 3.151088 4.650438 5.117128 6.457291
```

```
[2,] 2.990901 4.749001 4.770535 6.759048
[3,] 2.987780 4.876176 4.960417 7.068784
[4,] 2.857211 4.815935 4.522580 6.770756
[5,] 2.868259 4.887150 4.577038 6.860803
[6,] 3.088579 5.274816 4.651858 6.959012
```

```
> ## -> ok!
> P_hat <- cbind(1 / tmp, exp(fit[, , 1])/tmp, exp(fit[, , 2])/tmp)
> dim(P_hat)
```

```
[1] 4000   12
```

```
> tmp <- data.frame(sample = rep(1:nrow(P_hat), each = ncol(P_hat)),
+                   nd_line = rep(1:nrow(nd), ncol(df$y) * nrow(P_hat)),
+                   y = rep(rep(0:2, each = nrow(nd)), nrow(P_hat)))
> head(tmp, n = 13)
```

```
   sample nd_line y
1       1       1 0
2       1       2 0
3       1       3 0
4       1       4 0
5       1       1 1
6       1       2 1
7       1       3 1
8       1       4 1
9       1       1 2
10      1       2 2
11      1       3 2
12      1       4 2
13      2       1 0
```

```
> ## quick check:
> rbind(1:2, 3:4, 5:6)
```

```
     [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
```

```
> as.numeric(rbind(1:2, 3:4, 5:6))
```

```
[1] 1 3 5 2 4 6
```

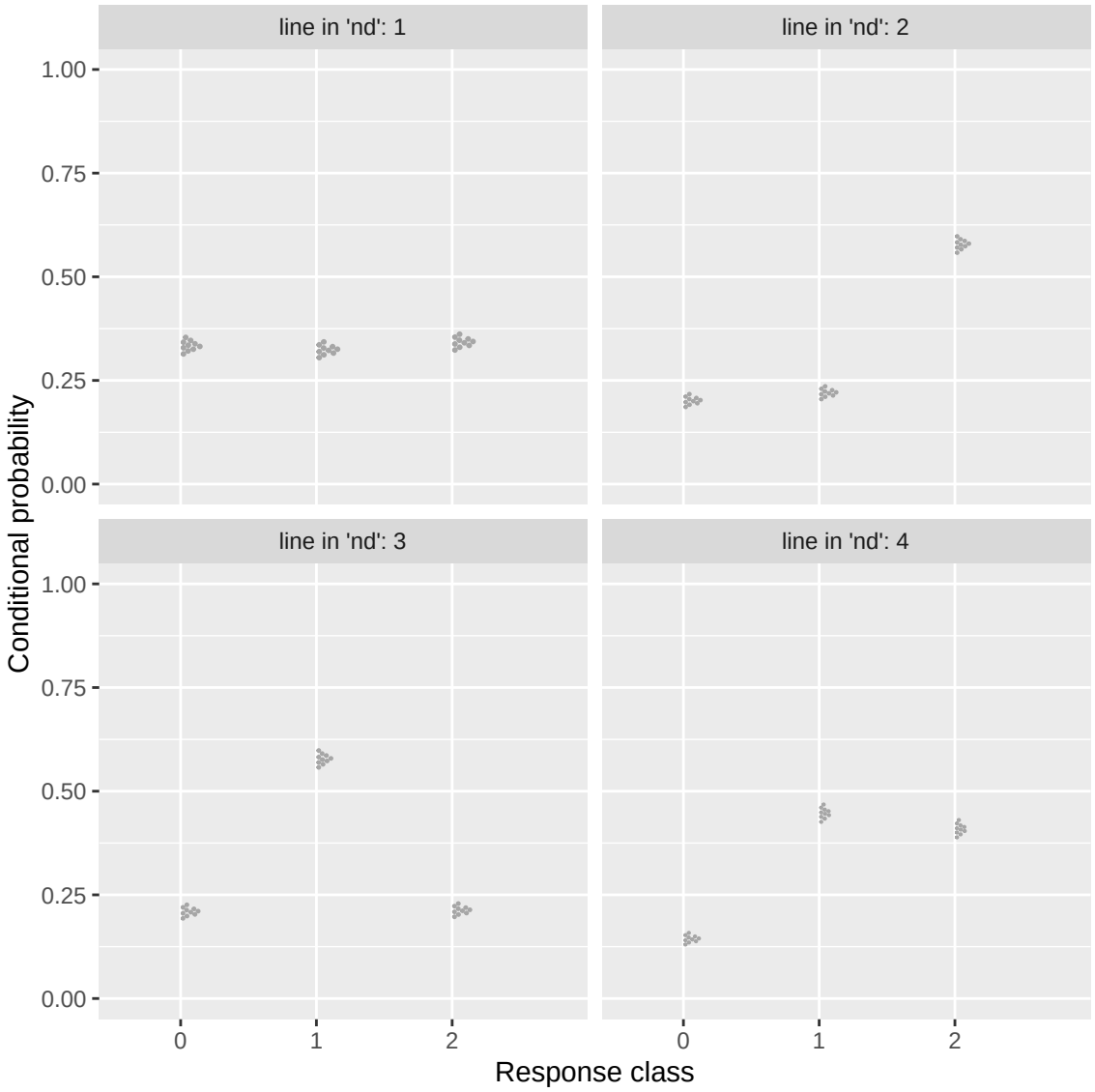
```
> as.numeric(t(rbind(1:2, 3:4, 5:6)))
```

```
[1] 1 2 3 4 5 6
```

```
> ## -> we need to transpose P_hat first.
> tmp$p <- as.numeric(t(P_hat))
> head(tmp)
```

```
  sample nd_line y         p
1      1       1 0 0.3173507
2      1       2 0 0.2150335
3      1       3 0 0.1954221
4      1       4 0 0.1548637
5      1       1 1 0.3198790
6      1       2 1 0.2034886
```

```
> ggplot(tmp, aes(x = as.factor(y), y = p)) +
+   stat_dots(quantiles = 10, layout = "swarm") +
+   facet_wrap(~ as.factor(paste0("line in 'nd': ", nd_line))) +
+   ylim(c(0, 1)) +
+   xlab(label = "Response class") +
+   ylab(label = "Conditional probability")
```




```
> str(fit)
```

```
 num [1:4000, 1:4, 1:2] 0.00794 -0.02313 -0.01315 -0.12605 -0.13296 ...
 - attr(*, "dimnames")=List of 3
  ..$ : NULL
  ..$ : NULL
  ..$ : NULL
```

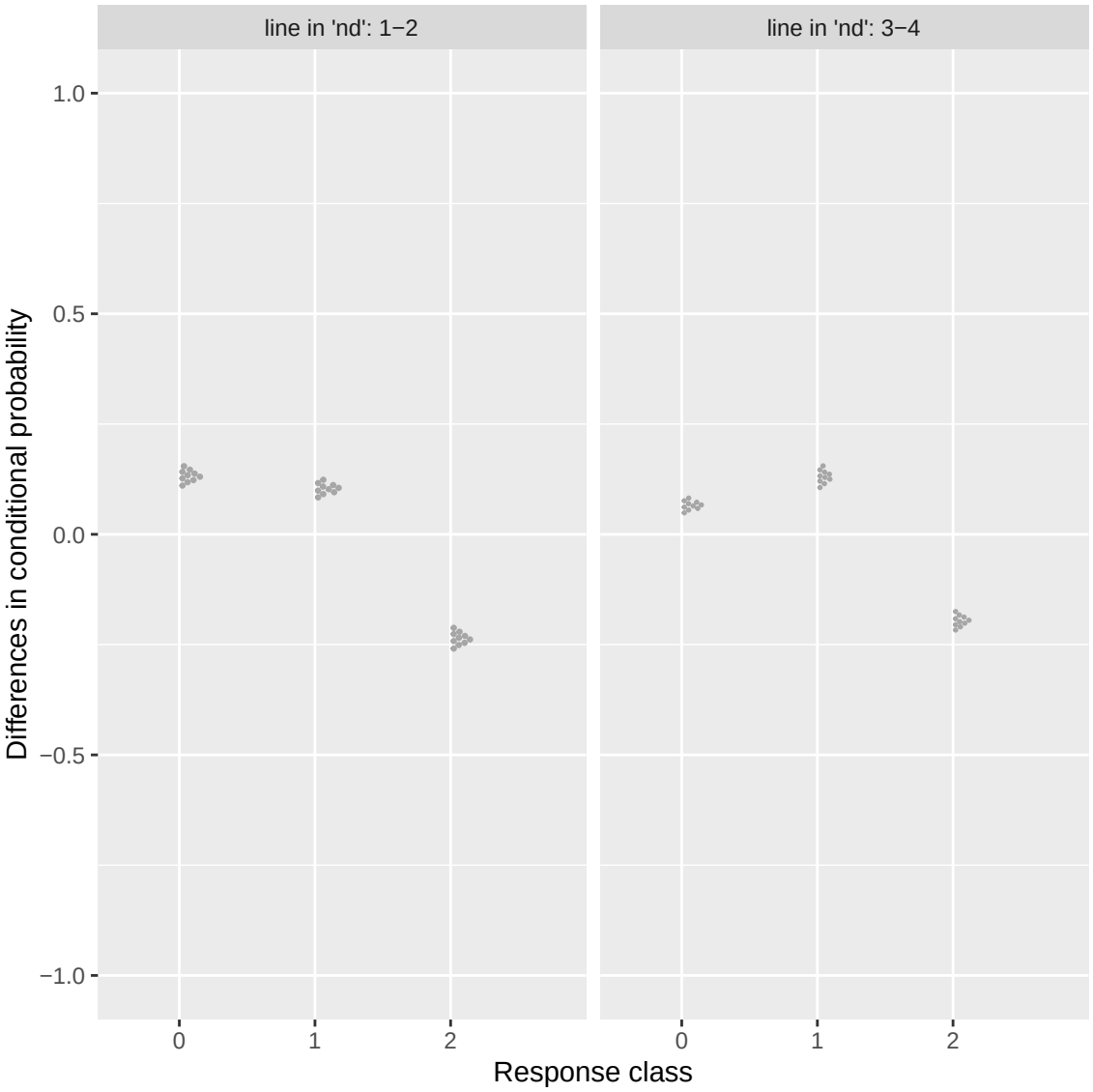
```
> summary(fit[, 1, 1] - fit[, 2, 1])
```

```
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-0.38946 -0.17123 -0.12007 -0.11943 -0.06754  0.20713
```

```
> summary(fit[, 3, 1] - fit[, 4, 1])
```

```
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-0.38946 -0.17123 -0.12007 -0.11943 -0.06754  0.20713
```

```
> summary(fit[, 1, 2] - fit[, 2, 2])

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-1.3152 -1.0846 -1.0301 -1.0302 -0.9781 -0.7244

> summary(fit[, 3, 2] - fit[, 4, 2])

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-1.3152 -1.0846 -1.0301 -1.0302 -0.9781 -0.7244

> (dd <- ddply(tmp, c("nd_line", "y"), summarise,
+              mean_p = mean(p)))

   nd_line y    mean_p
1        1 0 0.3337060
2        1 1 0.3238993
3        1 2 0.3423946
4        2 0 0.2013816
5        2 1 0.2202183
6        2 2 0.5784001
7        3 0 0.2094925
8        3 1 0.5777291
9        3 2 0.2127785
10       4 0 0.1439155
11       4 1 0.4470358
12       4 2 0.4090487

> subset(dd, nd_line == 1)$mean_p - subset(dd, nd_line == 2)$mean_p

[1]  0.1323244  0.1036810 -0.2360055

> subset(dd, nd_line == 3)$mean_p - subset(dd, nd_line == 4)$mean_p

[1]  0.06557699  0.13069324 -0.19627023
```


line in 'nd': 1−2
line in 'nd': 3−4
1.0
0.5
0.0
−0.5
−1.0
Differences in conditional probability
0
1
2
0
1
2
Response class

References

P.-C. Bürkner. brms: An R package for Bayesian multilevel models using Stan. *Journal of Statistical Software*, 80(1):1–28, 2017. doi: 10.18637/jss.v080.i01.

P.-C. Bürkner. Advanced Bayesian multilevel modeling with the R package brms. *The R Journal*, 10(1):395–411, 2018. doi: 10.32614/RJ-2018-017.

A. Eklund and J. Trimble. *beeswarm: The Bee Swarm Plot, an Alternative to Stripchart*, 2021. URL `https://CRAN.R-project.org/package=beeswarm`. R package version 0.4.0.

M. Kay. *ggdist: Visualizations of Distributions and Uncertainty*, 2021. URL `https://mjskay.github.io/ggdist/`. R package version 3.0.0.

P. Paolino. Predicted probabilities and inference with multinomial logit. *Political Analysis*, 29(3):416–421, Nov. 2020. doi: 10.1017/pan.2020.35. URL `https://doi.org/10.1017/pan.2020.35`.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2020. URL `https://www.R-project.org/`.

Stan Development Team. RStan: the R interface to Stan, 2021. URL `https://mc-stan.org/`. R package version 2.21.3.

H. Wickham. The split-apply-combine strategy for data analysis. *Journal of Statistical Software*, 40(1):1–29, 2011. URL `http://www.jstatsoft.org/v40/i01/`.

H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. ISBN 978-3-319-24277-4. URL `https://ggplot2.tidyverse.org`.