Piecewise Regression with Negative Binomial Type I Error on Real Data using `brms` Custom Family

Michael Gilchrist

Created: 2023-02-28; Compiled: Sat Mar 4 14:30:09 2023

Goal

- Fit two piece negative binomial type 1 formulation to data
- From 2023-02-28 version of `nbinom_type1.R`

  Negative Binomial distribution parameterized by mean (mu) and overdispersion parameter (theta). This parameterization is referred to as NEGBIN type I (Cameron and Trivedi, 1998) as cited by https://doi.org/10.1080/03610926.2018.1563164 ## `x ~ nbinom_type1(mu, theta)`, where `E(x) = mu, Var(x) = (theta + 1) mu` This should not be confused with the mu and shape parameterization of nbinom in R or the 'alternative' NB (`neg_binomial_2_...`) in stan Note using disp instead of theta because using theta gives the error > Error: Currently 'dirichlet' is the only valid prior for simplex parameters. See help(set_prior) for more details. when trying to fit the model.

Recap

- Code initially based on work in
- `../2022-12-20_rstan_two.piece.qpoisson.with.real.data/rstan_two.piece_fit.various.models.to.real.da`
-
- That model behaved well but, because I was writing stan code directly, I hadn't been able to group data by `male`
- Since then, I've created a `brms custom_family()` which uses the Type 1 formulation of the NB.

Insights

- When the `disp` (dispersal or `theta`) gets unrealistically large, we get the emergence of a bi modal distribution at both ends of `x0` values.
  Even though we included this value, it is very unlikely to be 25C values. I interpret this to mean that when things are really noisy (high `theta`), one way to interpret the data is that one set of males has a very long (presumably slow) decline. It would be good to look at the correlations via `pairs()`.
- To me this is consistent with the infomal knowledge that the

Set up

Install libraries

```
# install packages user might not have by replacing FALSE with TRUE

## load libraries
library(stats)
```

```
library(MASS) # provides negative binomial fitting:  glm.nb
library(ggplot2)
library(ggpubr)
library(grid)
library(gridExtra)
library(GGally)
library(broom)
library(tidyverse)
library(viridisLite)
library(cmdstanr)
library(rstan)
options(mc.cores = (parallel::detectCores()-2))
rstan_options(auto_write = TRUE)
library(brms)
library(loo)


## options(ggplot2.continuous.colour="viridis",
##         ggplot2.discrete.colour="viridis",
##         ggplot2.scale_fill_discrete = scale_fill_viridis_d,
##         ggplot2.scale_fill_continuous = scale_fill_viridis_c)

library(reshape2)
library(lme4)
library(latex2exp)
```

Source family

```
source("../../../custom-brms-families/families/nbinom_type1.R")
```

```
## Error in validate_stanvars(stanvars, stan_funs = stan_funs): object 'stanvars' not found
```

Load Data

```
sapply(file.path("input", dir("input")),
       load, verbose = TRUE, envir = .GlobalEnv)
```

```
## Loading objects:
##   data_ind
## Loading objects:
##   summary_stats
## Loading objects:
##   stats_ind
```

```
##          input/data_ind.Rda input/obs_summary_stats.Rda
##                  "data_ind"             "summary_stats"
##         input/stats_ind.Rda
##                 "stats_ind"
```

```
head(stats_ind)
```

```
## # A tibble: 6 x 9
##   male  round n_obs total_round mean_round sd_round cv_round total  mean
##   <fct> <dbl> <int>       <int>      <dbl>    <dbl>    <dbl> <int> <dbl>
```

```
## 1 T234      1    13         203       40.6     32.0    0.787   601  46.2
## 2 T235      1    13         882      176.     132.     0.748  2333 179.
## 3 T236      1    13         758      152.      46.0    0.303  2095 161.
## 4 T243      1    13         438       87.6     76.4    0.872  1861 143.
## 5 T244      1    13         270       54       14.7    0.272   993  76.4
## 6 T246      1     5         253       50.6     54.6    1.08    253  50.6
```

```
names(stats_ind)
```

```
## [1] "male"        "round"       "n_obs"       "total_round" "mean_round"
## [6] "sd_round"    "cv_round"    "total"       "mean"
```

```
head(data_ind)
```

```
## # A tibble: 6 x 11
##   male  index motif_~1 temp_~2  temp round trial~3 date  counter y0_si~4 phi_ind
##   <chr> <int>    <int>   <dbl> <dbl> <dbl>   <dbl> <chr> <chr>     <dbl>   <dbl>
## 1 T234      1        0      42  43.0     1       1 02/0~ RAS        46.2    12.3
## 2 T234      1       30      44  44.5     1       2 02/0~ RAS        46.2    12.3
## 3 T234      1       34      27  27.2     1       3 02/0~ RAS        46.2    12.3
## 4 T234      1       87      40  41.1     1       4 02/0~ RAS        46.2    12.3
## 5 T234      1       52      35  36.1     1       5 02/1~ RAS        46.2    12.3
## 6 T234      1       32      40  39.5     2       1 04/2~ KIM        46.2    12.3
## # ... with abbreviated variable names 1: motif_count, 2: temp_target,
## #   3: trial_round, 4: y0_simple_est
```

```
names(data_ind)
```

```
##  [1] "male"          "index"         "motif_count"   "temp_target"
##  [5] "temp"          "round"         "trial_round"   "date"
##  [9] "counter"       "y0_simple_est" "phi_ind"
```

Prior Work

Fit with Stan: Separate: `y0`, and `x0`, Pooled θ

- Based on results in `../2022-12-20_rstan_two.piece.qpoisson.with.real.data/rstan_two.piece_fit.various.m`
- Histograms of `x0[]` suggest following groupings

1. `c(2, 4, 6, 8, 11)`
2. `c(1, 3, 5, 7, 10)`
3. `9` is ambiguous.

- NOTE: I'm not 100% sure these index values are the same as the ones we are using here!

```
x0_groups_2_base <- list(low = c(2, 4, 6, 8, 11), high = c(1, 3, 5, 7, 10)) ## Does not include bird wi

x0_groups_2a <- x0_groups_2b <- x0_groups_2_base

x0_groups_2a$low <- c(x0_groups_2_base$low, 9) %>% sort()
x0_groups_2b$high <- c(x0_groups_2_base$high, 9) %>% sort()
```

Fit Models

- Code derived from `../2023-02-23_brms_nb_focus.on.x0.fittings/nb_focus.on.estimating.x0.fittings.Rmd`

Set up functions, parameters, and results tibble

```
data_stan <- data_ind %>% rename(y = motif_count, x = temp) %>%
  mutate(male = factor(male))

xmax <- 46
stan_two_piece_func <- paste0(" real  two_piece(real x, real x0, real y0) {
 real xmax = ", xmax, "; ## paste in value for xmax
 ## return y0 * (xmax - fmax(x0, x))/(xmax - x0);
 real val1 = xmax  - x;
 real val2 = xmax - x0;
 real b0 = -y0/(xmax - x0);
 real y = y0 - fmin(val1, val2)*b0;
 return(y);
 } ")

models <- c("piecewise") #, "asymptotic")
sampling_dists <- c("nbinom_type1") ##, "com_poisson") ## lognormal doesn't work since the counts can b
flags_x0 <- c("uniform_1",
              "groups_1", ## this doesn't work with x0_Intercept prior, suggests error in priors
              "groups_2a",
              "groups_2b",
              "individual")

flags_y0 <- c("uniform_1", "groups_1", "individual")

disp_prior_list <- c(0.5, 2, 4, 16, 64, 128)

N <- length(data)

fit_tbl <- crossing(model = models,
                    sampling_dist = sampling_dists,
                    x0_flag = flags_x0, y0_flag = flags_y0,
                    disp_prior = disp_prior_list,
                    desc = "NA_character",
                    y0_group_list = list(NA), #tbl_tmp, #list(NA),
                    x0_group_list = list(NA),
                    fit = list(NA),
                    llik = list(NA),
                    r_eff = list(NA),
                    loo = list(NA)
                    )
```

Run fits

```
run_fits = TRUE

sampling = "nbinom_type1"
flag_category <- "individual_individual"


switch(flag_category,
       simple = { ## x0 and y0 have single, shared value
         flags_x0_used = "uniform_1"
```

```
        flags_y0_used = "uniform_1"
      },
      simple_individual = {
        flags_x0_used = "uniform_1"
        flags_y0_used = "individual"
      },
      group1_individual ={
        flags_x0_used = "groups_1"
        flags_y0_used = "individual"
      },
      individual_individual = { # values vary between males
        flags_x0_used = "individual"
        flags_y0_used = "individual"
      }
      )

flags_x0_used = c("individual", "uniform_1", "group_1")
flags_y0_used = c("individual")

fit_index <- 0;

for(model in models[[1]]) {

  switch(sampling,
         "nbinom_type1"= {
           family <- nbinom_type1(link = "identity", link_disp = "log")
           adapt_delta <- 0.8 #0.95 ## will decreasing value increase ESS? Seems like it
           iter <- 50000 # could try <100000
           warmup <- floor(1/2 * iter)
           thin <- 10
           n_cores <- 4 ## set to 1 if getting errors from stan in order to see relevant message.
           n_chains <- n_cores
           nbinom_type1_vars <-
             stanvar(scode = paste(
               stan_two_piece_func,
               stan_nbinom_type1, sep = "\n"),
               block = "functions")
         }
         )


  for(x0_flag in flags_x0_used) {
    for(y0_flag in flags_y0_used) {
      for(disp_prior in disp_prior_list) {

        ## used when loading fits
        fit_index <- fit_index + 1;
        ## Set up variables for saving model and fit
        desc <- paste0(model, "; ", sampling, "; x0 " , x0_flag, "; y0 ", y0_flag, ", disp prior: ", dis
        curr_row <- which(fit_tbl$sampling_dist == sampling &
                            fit_tbl$x0_flag == x0_flag &
                            fit_tbl$y0_flag == y0_flag &
                            fit_tbl$disp_prior == disp_prior)
```

```
fit_tbl[ curr_row, ]$desc <- desc

print(desc)

outfile_tbl <- file.path(output_dir, paste0("fit_tbl_", Sys.Date() , ".Rda"))
if(run_fits) {
## Refresh data in case x0_group or y0_group are all set to 1
data <- data_stan

## Set flags based on fitted model structure
if(x0_flag %in% c("uniform_1", "groups_1")) data <- mutate(data, x0_group = 1)
if(y0_flag %in% c("uniform_1", "groups_1")) data <-mutate(data, y0_group = 1)
if(x0_flag %in% c("individual")) data <- mutate(data, x0_group = male)
if(y0_flag %in% c("individual")) data <- mutate(data, y0_group = male)

## Note we need to put a tibble into a list because row updates, even if doing
## just one cell, require a list format.
  ## Haven't defined "y0_group" or "x0_group" variables
  if(FALSE) {
    fit_tbl[[curr_row, "x0_group_list"]] <- list(unique(data[, c("male", "x0_group")]))
    fit_tbl[[curr_row, "y0_group_list"]] <- list(unique(data[, c("male", "y0_group")]))
  }

## Parameter Structure
x0_form <- switch(x0_flag,
                  uniform_1 = formula(x0 ~ 1),
                  uniform_2 = formula(x0 ~ x0_group),
                  # Don't include x0_group info which is determined by the data set
                  groups_1 = formula(x0 ~ 1 + (1|male)),
                  groups_2 = formula(x0 ~ 1 + (1|male) + x0_group),
                  individual = formula(x0 ~ 0 + male) ## Do not use 1 + male!
                  )

y0_form <- switch(y0_flag,
                  uniform_1 = formula(y0 ~ 1),
                  uniform_2 = formula(y0 ~ 0 + 1 + y0_group),
                  groups_1 = formula(y0 ~ 0 + 1 + (1| male)),
                  groups_2 = formula(y0 ~ 0 + 1 + (1| male) + y0_group),
                  individual = formula(y0 ~ 0 + male)
                  ## `-1 + ` gives me the error:
                  ## Warning in parallel::mclapply(1:chains, FUN = callFun, mc.preschedule = FA
                  ##  4 function calls resulted in an error
                  ## Error in FUN(X[[i]], ...) :
                  ##  trying to get slot "mode" from an object (class "try-error") that is not
                  ## should I use `0 +` or `-1 +` ?
                  )

## Priors
## Strangely reducing the parameter of the exponential results in a higher estimate of 'disp' a
prior_core <- switch(1,
                     set_prior( paste0("exponential(", disp_prior, ")"), class = "disp", lb = 0
                     normal(15, 2.8), ## Based on analysis of non-noisy 40C song data
                     normal(6, 2.8), ## Based on analysis of non-noisy 40C song data
```

```
                        exponential(0.667), ## 0.625 = 1/16  "
                        constant(5) ## Making disp a constant oesn't really help.
                        ) +
  prior(uniform(30, 45), lb = 30, ub = 45, nlpar = "x0")

## Parameter Structure
x0_priors <- switch(x0_flag,
                    uniform_1 = ,
                    uniform_2 = ,
                    # Don't include x0_group info which is determined by the data set
                    groups_1 = ,
                    groups_2 = ,
                    individual = ,
                    )

y0_priors <- switch(y0_flag,
                    uniform_1 = prior(normal(150, 100), nlpar = "y0"),
                    uniform_2 = ,
                    groups_1 = ,
                    groups_2 = ,
                    individual = prior(normal(150, 1000), nlpar = "y0", lb = 10, ub = 1000)
                    )

prior <- switch(sampling,
                "nbinom_type1" = {
                  prior_core +
                    x0_priors + y0_priors
                })

nlform <- bf(
  y ~ two_piece(x, x0, y0), nl = TRUE) +
  x0_form +
  y0_form

if(FALSE) print(get_prior(nlform,
                          data = data,
                          family = family
                          ))

  save_model <-paste0(paste0(output_dir, "/stan/brms", model, sampling, x0_flag, y0_flag, disp_
#make_stancode( .... save_model = save.model)
fit <- brm(nlform,
           data = data,
           ## `link` refers to the mapping of the expectation of the distribution: log, sqrt, i
           ## link_shape corresponds to `phi` of `stan`'s
           ## Negbinomial_2
           ## Defining `phi = mu/theta` creates a quasipoisson
           ## distribution with overdispersion parameter (1 +theta)
           family = family, #negbinomial(link = "identity", link_shape = "identity"),
           prior = prior,
           stanvars = nbinom_type1_vars,
           iter = iter,
           warmup = warmup,
```

```
                    thin = thin,
                    silent = ifelse(interactive(), 1, 2), # 0, 1, or 2. 1 is default
                    control = list(adapt_delta = adapt_delta,
                                   max_treedepth = 12
                                   ##model_name = desc ## Incorrect way to set this.
                                   ),
                    ## Only print out sampling progress if in interactive mode
                    refresh = ifelse(interactive(),max(iter/5, 1), 0),
                    chains = n_chains,
                    cores = n_cores,
                    save_model = save_model
                    )

          #fit_exp <- expose_functions(fit) , vectorize = TRUE)
            #fit_cr <- add_criterion(fit_exp, c("loo", "waic"))
            fit_tbl[[curr_row, "fit"]] <- list(fit)
            ## Print current warnings
            warnings(summary)
            ## Clear warnings()

            ## End if(run_fit)
          } else {
            if(fit_index ==1)  load(file = outfile_tbl)
            #load from local memory.
            fit <- fit_tbl[[curr_row, "fit"]][[1]]


            print(desc)
            print("Prior Information")
            print(prior_summary(fit))
            print("Fit Information")
            print(fit)
            fit_stan <- fit$fit

            ##warning(immediate. = FALSE)
            hist <- stan_hist(fit_stan, pars = c("b_y0", "disp"), main = desc )
            print(hist)
            ## Save hist to file
            hist <- stan_hist(fit_stan, pars = c("b_x0", "disp"), main = desc)
            print(hist)
            ## Save hist to file
          }  #end else for fitting
        }
      }
    }
  }
}
```

```
## [1] "piecewise; nbinom_type1; x0 individual; y0 individual, disp prior: 0.5"
## recompiling to avoid crashing R session
## [1] "piecewise; nbinom_type1; x0 individual; y0 individual, disp prior: 2"
## recompiling to avoid crashing R session
## [1] "piecewise; nbinom_type1; x0 individual; y0 individual, disp prior: 4"
```

```
## recompiling to avoid crashing R session
## [1] "piecewise; nbinom_type1; x0 individual; y0 individual, disp prior: 16"
## recompiling to avoid crashing R session
## [1] "piecewise; nbinom_type1; x0 individual; y0 individual, disp prior: 64"
## recompiling to avoid crashing R session
## Warning: The largest R-hat is 1.5, indicating chains have not mixed.
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#r-hat
## Warning: Bulk Effective Samples Size (ESS) is too low, indicating posterior means and medians may be
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#bulk-ess
## Warning: Tail Effective Samples Size (ESS) is too low, indicating posterior variances and tail quanti
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#tail-ess
## [1] "piecewise; nbinom_type1; x0 individual; y0 individual, disp prior: 128"
## recompiling to avoid crashing R session
## Warning: The largest R-hat is 1.47, indicating chains have not mixed.
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#r-hat
## Warning: Bulk Effective Samples Size (ESS) is too low, indicating posterior means and medians may be
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#bulk-ess
## Warning: Tail Effective Samples Size (ESS) is too low, indicating posterior variances and tail quanti
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#tail-ess
## [1] "piecewise; nbinom_type1; x0 uniform_1; y0 individual, disp prior: 0.5"
## recompiling to avoid crashing R session
## [1] "piecewise; nbinom_type1; x0 uniform_1; y0 individual, disp prior: 2"
## [1] "piecewise; nbinom_type1; x0 uniform_1; y0 individual, disp prior: 4"
## [1] "piecewise; nbinom_type1; x0 uniform_1; y0 individual, disp prior: 16"
## [1] "piecewise; nbinom_type1; x0 uniform_1; y0 individual, disp prior: 64"
## [1] "piecewise; nbinom_type1; x0 uniform_1; y0 individual, disp prior: 128"
## [1] "piecewise; nbinom_type1; x0 group_1; y0 individual, disp prior: 0.5"
## Error: The following variables can neither be found in 'data' nor in 'data2':
## 'x0'
print(outfile_tbl)
## [1] "output/render/fit_tbl_2023-03-04.Rda"
if(run_fits) save(fit_tbl, file = outfile_tbl)
```