Piecewise Regression QPoisson Error on Real Data using STAN Directly

Michael Gilchrist

date: 2022-12-22

Goal

- Fit two piece quasipoisson to data

Recap

Set up

Install libraries

```r
# 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(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)
```

Load Data

```
load(file.path("input", "data.processing_2022-12-15.Rda"),
     verbose = TRUE)
```

```
## Loading objects:
##   motif_data
##   motif_data_40C
##   motif_stats
##   motif_stats_40C
##   bird_bill_data
```

```
motif_data
```

```
## # A tibble: 146 x 28
##    male  round trial_round motif~1 motif~2 temp_~3 humid~4 chamber date  counter
##    <fct> <dbl>       <dbl>   <int>   <dbl>   <dbl>   <dbl> <fct>   <chr> <chr>
##  1 T229      1           1       0 0          45.8      NA 6       02/1~ KIM
##  2 T229      1           2      24 0.0131     42.3      NA 6       02/1~ KIM
##  3 T229      1           3     114 0.0622     40.7      NA 6       02/1~ KIM
##  4 T229      1           4     198 0.108      26.2      NA 6       02/1~ KIM
##  5 T229      1           5     315 0.172      34.9      NA 6       02/2~ KIM
##  6 T231      1           1      57 0.0431     42.8      NA 2       02/1~ RAS
##  7 T231      1           2       7 0.00529    45.0      NA 2       02/1~ RAS
##  8 T231      1           3      86 0.0650     41.1      NA 2       02/1~ KIM
##  9 T231      1           4      24 0.0181     27.2      NA 2       02/1~ RAS
## 10 T231      1           5     215 0.162      36.5      NA 2       02/2~ RAS
## # ... with 136 more rows, 18 more variables: test_order <int>,
## #   temp_target <dbl>, temp_median <dbl>, humdity_mean <dbl>, motif_rate <dbl>,
## #   mass <dbl>, n_obs_completed <lgl>, motif_count_plus_1 <int>,
## #   log_motif_count_plus_1 <dbl>, temp <dbl>, n_obs_round <int>, n_obs <int>,
## #   trial <int>, motif_prop_round <dbl>, weights <dbl>, svp <dbl>, vpd <dbl>,
## #   vpd_offset <dbl>, and abbreviated variable names 1: motif_count,
## #   2: motif_prop, 3: temp_mean, 4: humidity_mean
```

Examine Data

Create Working Dataset

```
filter_data <- TRUE

if(filter_data) {
  males_filtered_disp <- motif_stats_40C %>%
    filter(dispersion < 50) %>%
```

```
    pull(male)

  males_filtered_mean <- motif_stats %>%
    filter(mean > 10) %>% # changing from 10 to 40 removes previous male 7 (T258)
    pull(male)

  male_vector <- intersect(males_filtered_mean, males_filtered_disp)
} else {
  male_vector <- motif_data %>% select(male) %>% distinct()
}

data_ind <- motif_data %>%
    filter(male %in% male_vector) %>%
    mutate(male = droplevels(male)) %>%
    mutate(index = as.integer(male)) %>%
    mutate(male = as.character(male)) %>%
    arrange(index) %>%
    select(male, index, motif_count, temp, round, trial_round, date, counter) %>%
  ##    left_join(index_shape, by = "index") %>%
  mutate()

stats_ind <- motif_stats %>%
    filter(male %in% male_vector)


data_ind <- data_ind %>% filter(temp < 38) %>%
    group_by(male) %>% mutate(y0_simple.est = mean(motif_count), phi_ind = var(motif_count)/y0_simple.es
    ungroup()

summary(data_ind)
```

```
##      male               index         motif_count         temp      
##  Length:38          Min.   : 1.000   Min.   :  0.0   Min.   :25.71  
##  Class :character   1st Qu.: 3.000   1st Qu.: 52.5   1st Qu.:29.51  
##  Mode  :character   Median : 5.500   Median : 89.0   Median :33.52  
##                     Mean   : 5.579   Mean   :112.9   Mean   :31.88  
##                     3rd Qu.: 8.000   3rd Qu.:167.2   3rd Qu.:34.34  
##                     Max.   :11.000   Max.   :425.0   Max.   :37.64  
##      round        trial_round        date             counter         
##  Min.   :1.000   Min.   :1.000   Length:38          Length:38         
##  1st Qu.:1.000   1st Qu.:2.000   Class :character   Class :character  
##  Median :3.000   Median :3.000   Mode  :character   Mode  :character  
##  Mean   :2.211   Mean   :3.105                                        
##  3rd Qu.:3.000   3rd Qu.:4.000                                        
##  Max.   :3.000   Max.   :6.000                                        
##  y0_simple.est       phi_ind       
##  Min.   : 24.00   Min.   :  0.142  
##  1st Qu.: 64.33   1st Qu.:  6.955  
##  Median : 93.20   Median : 12.986  
##  Mean   :112.89   Mean   : 33.913  
##  3rd Qu.:166.38   3rd Qu.: 48.000  
##  Max.   :246.25   Max.   :128.361  
```

```r
summary_stats <- data_ind %>% ungroup() %>%  summarize(y0_bar = mean(y0_simple.est), y0_sd = sd(y0_simpl

n_male <- length(unique(data_ind$male))
```

Set Up Data

```r
data <- data_ind
motif_count <- data %>% pull(motif_count)
temp <- data %>% pull(temp)
N <- length(temp)
index <- data %>% pull(index)
## parameters to be printed

pars <- c("t0", "y0")
pars_full <- c(pars, "lp__")
```

Fit Models

QPoisson

```r
iter <- 15000
tmax <- 46
t0max <- tmax - 0.5;
t0min <- 20;
## values to use for model predictions
tp = seq(25, tmax, length.out = 100)
n_cores <- 4
n_chains <- n_cores

##y0_grouping <- map_int(data$male, ~ if_else(. %in% y0_group[[1]], 1, 2))
model <- "qpoi"
stan_file <- "two.piece_qpoisson_2.0.stan"
## For debugging
## cmodel <- cmdstan_model(stan_file = stan_file)

stan_model(file = stan_file,
           verbose = TRUE)
```

```
##
## TRANSLATING MODEL '' FROM Stan CODE TO C++ CODE NOW.
```

```r
## Define groups

flags <- c("separate", "grouping_1", "pooled")
flags_x <- flags
flags_y <- flags
fit_tbl <- crossing(model = model,
                    x0 = flags_x, y0 = flags_y,
```

```
                    desc = "NA_character",
                    y0_group_list = list(NA),
                    x0_group_list = list(NA),
                    fit = list(NA),
                    llik = list(NA),
                    r_eff = list(NA),
                    loo = list(NA)
                    )



for(x_flag in flags_x) {
    for(y_flag in flags_y) {

        desc <- paste0(model, ": ", x_flag, ", ", y_flag)
        curr_row <- which(fit_tbl$x0 == x_flag & fit_tbl$y0 == y_flag)

        fit_tbl[ curr_row, ]$desc <- desc
        print(desc)

        x0_group_list <- list()
        y0_group_list <- list()

        switch(x_flag,
               separate = {
                   x0_group_list <- data$male %>% unique() %>% as.list()
               },
               grouping_1 = {
                   ## set up groupings based on 2022-12-20 analysis
                   ## Using male ID's instead index to make code more robust
                   x0_group_list[[1]] <- c("T235", "T237", "T244", "T247", "T257", "T260")
                   x0_group_list[[2]] <- c("T234", "T236", "T243", "T246", "T258")
               },
               pooled = {
                   x0_group_list[[1]] <- data$male
               }
               )

        switch(y_flag,
               separate = {
                   y0_group_list <- data$male %>% unique() %>% as.list()
               },
               grouping_1 = {
                   ## set up groupings based on 2022-12-20 analysis
                   ## Using male ID's instead index to make code more robust
                   y0_group_list[[1]] <- c("T234", "T243", "T244", "T246", "T258", "T260")
                   y0_group_list[[2]] <- c("T235", "T236", "T237", "T247", "T257")
               },
               pooled = {
                   y0_group_list[[1]] <- data$male
               }
               )
```

```r
        fit_tbl[ curr_row, ]$x0_group_list[[1]] <- x0_group_list

        fit_tbl[ curr_row, ]$y0_group_list[[1]] <- y0_group_list

        ## Convert lists to a vector of concatenated strings
        ## This will simplify mapping male to an x0/y0 index
        x0_group <-lapply(x0_group_list, paste, collapse = " ") %>% unlist()
        y0_group <-lapply(y0_group_list, paste, collapse = " ") %>% unlist()

        x0_index <- sapply(as.character(data$male), function(x) str_which(x0_group, x))
        y0_index <- sapply(as.character(data$male), function(x) str_which(y0_group, x))

        fit <- stan(file = stan_file,
                    model_name = desc,
                    data=list(x = temp,
                              y = motif_count,
                              N = N,
                              X = length(x0_group),
                              Y = length(y0_group),
                              NB = 1,
                              xx = x0_index,
                              yy = y0_index,
                              nbb = rep(1,N),
                              xmax = tmax,
                              x0_min = t0min,
                              x0_max = t0max,
                              y_xmax = 0,
                              y0_min = 10,
                              sd_y0_prior = 200,
                              alpha_theta_prior = 10,
                              ##tp = tp,
                              ## max threshold value.
                              ## having it too close to xmax *sometimes* leads to sampling
                              ## near xmax, but with lower lp and very high E13) b0 values
                              y_xmax = 0),
                  cores = n_cores,
                  chains = n_chains,
                  iter = iter,
                  warmup = floor(iter/2),
                  verbose = TRUE)

        fit_tbl[ curr_row, ]$fit <- list(fit)

    }
}
```

```
## [1] "qpoi: separate, separate"
##
## TRANSLATING MODEL 'qpoi: separate, separate' FROM Stan CODE TO C++ CODE NOW.
##
## CHECKING DATA AND PREPROCESSING FOR MODEL 'anon_model' NOW.
##
## COMPILING MODEL 'anon_model' NOW.
```

```
##
## STARTING SAMPLER FOR MODEL 'anon_model' NOW.
## [1] "qpoi: separate, grouping_1"
##
## TRANSLATING MODEL 'qpoi: separate, grouping_1' FROM Stan CODE TO C++ CODE NOW.
##
## CHECKING DATA AND PREPROCESSING FOR MODEL 'anon_model' NOW.
##
## COMPILING MODEL 'anon_model' NOW.
##
## STARTING SAMPLER FOR MODEL 'anon_model' NOW.
## [1] "qpoi: separate, pooled"
##
## TRANSLATING MODEL 'qpoi: separate, pooled' FROM Stan CODE TO C++ CODE NOW.
##
## CHECKING DATA AND PREPROCESSING FOR MODEL 'anon_model' NOW.
##
## COMPILING MODEL 'anon_model' NOW.
##
## STARTING SAMPLER FOR MODEL 'anon_model' NOW.
## [1] "qpoi: grouping_1, separate"
##
## TRANSLATING MODEL 'qpoi: grouping_1, separate' FROM Stan CODE TO C++ CODE NOW.
##
## CHECKING DATA AND PREPROCESSING FOR MODEL 'anon_model' NOW.
##
## COMPILING MODEL 'anon_model' NOW.
##
## STARTING SAMPLER FOR MODEL 'anon_model' NOW.
## [1] "qpoi: grouping_1, grouping_1"
##
## TRANSLATING MODEL 'qpoi: grouping_1, grouping_1' FROM Stan CODE TO C++ CODE NOW.
##
## CHECKING DATA AND PREPROCESSING FOR MODEL 'anon_model' NOW.
##
## COMPILING MODEL 'anon_model' NOW.
##
## STARTING SAMPLER FOR MODEL 'anon_model' NOW.
## [1] "qpoi: grouping_1, pooled"
##
## TRANSLATING MODEL 'qpoi: grouping_1, pooled' FROM Stan CODE TO C++ CODE NOW.
##
## CHECKING DATA AND PREPROCESSING FOR MODEL 'anon_model' NOW.
##
## COMPILING MODEL 'anon_model' NOW.
##
## STARTING SAMPLER FOR MODEL 'anon_model' NOW.
## [1] "qpoi: pooled, separate"
##
## TRANSLATING MODEL 'qpoi: pooled, separate' FROM Stan CODE TO C++ CODE NOW.
##
## CHECKING DATA AND PREPROCESSING FOR MODEL 'anon_model' NOW.
##
## COMPILING MODEL 'anon_model' NOW.
```

```
##
## STARTING SAMPLER FOR MODEL 'anon_model' NOW.
## [1] "qpoi: pooled, grouping_1"
##
## TRANSLATING MODEL 'qpoi: pooled, grouping_1' FROM Stan CODE TO C++ CODE NOW.
##
## CHECKING DATA AND PREPROCESSING FOR MODEL 'anon_model' NOW.
##
## COMPILING MODEL 'anon_model' NOW.
##
## STARTING SAMPLER FOR MODEL 'anon_model' NOW.
## [1] "qpoi: pooled, pooled"
##
## TRANSLATING MODEL 'qpoi: pooled, pooled' FROM Stan CODE TO C++ CODE NOW.
##
## CHECKING DATA AND PREPROCESSING FOR MODEL 'anon_model' NOW.
##
## COMPILING MODEL 'anon_model' NOW.
##
## STARTING SAMPLER FOR MODEL 'anon_model' NOW.
```

```
## save(file = "fit_tbl.Rda", fit_tbl)
qpoisson_fit_tbl <- fit_tbl
```

- Models fit without any warnings.

Negative Binomial

```
iter <- 15000
tmax <- 46
t0max <- tmax - 0.5;
t0min <- 20;
## values to use for model predictions
tp = seq(25, tmax, length.out = 100)
n_cores <- 4
n_chains <- n_cores

##y0_grouping <- map_int(data$male, ~ if_else(. %in% y0_group[[1]], 1, 2))
model <- "nb"
stan_file <- "two.piece_nb_1.0.stan"
## For debugging
##cmodel <- cmdstan_model(stan_file = stan_file)

stan_model(file = stan_file,
           verbose = TRUE)
```

```
##
## TRANSLATING MODEL '' FROM Stan CODE TO C++ CODE NOW.
```

```r
## Define groups

flags <- c("separate", "grouping_1", "pooled")
flags_x <- flags
flags_y <- flags
fit_tbl <- crossing(model = model,
                    x0 = flags_x, y0 = flags_y,
                    desc = "NA_character",
                    y0_group_list = list(NA),
                    x0_group_list = list(NA),
                    fit = list(NA),
                    llik = list(NA),
                    r_eff = list(NA),
                    loo = list(NA)
                    )



for(x_flag in flags_x) {
    for(y_flag in flags_y) {

        desc <- paste0(model, ": ", x_flag, ", ", y_flag)
        curr_row <- which(fit_tbl$x0 == x_flag & fit_tbl$y0 == y_flag)

        fit_tbl[ curr_row, ]$desc <- desc
        print(desc)

        x0_group_list <- list()
        y0_group_list <- list()

        switch(x_flag,
               separate = {
                   x0_group_list <- data$male %>% unique() %>% as.list()
               },
               grouping_1 = {
                   ## set up groupings based on 2022-12-20 analysis
                   ## Using male ID's instead index to make code more robust
                   x0_group_list[[1]] <- c("T235", "T237", "T244", "T247", "T257", "T260")
                   x0_group_list[[2]] <- c("T234", "T236", "T243", "T246", "T258")
               },
               pooled = {
                   x0_group_list[[1]] <- data$male
               }
               )

        switch(y_flag,
               separate = {
                   y0_group_list <- data$male %>% unique() %>% as.list()
               },
               grouping_1 = {
                   ## set up groupings based on 2022-12-20 analysis
                   ## Using male ID's instead index to make code more robust
                   y0_group_list[[1]] <- c("T234", "T243", "T244", "T246", "T258", "T260")
```

```
                y0_group_list[[2]] <- c("T235", "T236", "T237", "T247", "T257")
            },
            pooled = {
                y0_group_list[[1]] <- data$male
            }
            )

        fit_tbl[ curr_row, ]$x0_group_list[[1]] <- x0_group_list

        fit_tbl[ curr_row, ]$y0_group_list[[1]] <- y0_group_list

        ## Convert lists to a vector of concatenated strings
        ## This will simplify mapping male to an x0/y0 index
        x0_group <-lapply(x0_group_list, paste, collapse = " ") %>% unlist()
        y0_group <-lapply(y0_group_list, paste, collapse = " ") %>% unlist()

        x0_index <- sapply(as.character(data$male), function(x) str_which(x0_group, x))
        y0_index <- sapply(as.character(data$male), function(x) str_which(y0_group, x))

        fit <- stan(file = stan_file,
                     model_name = desc,
                     data=list(x = temp,
                               y = motif_count,
                               N = N,
                               X = length(x0_group),
                               Y = length(y0_group),
                               NB = 1,
                               xx = x0_index,
                               yy = y0_index,
                               nbb = rep(1,N),
                               xmax = tmax,
                               x0_min = t0min,
                               x0_max = t0max,
                               y_xmax = 0,
                               y0_min = 10,
                               sd_y0_prior = 200,
                               alpha_theta_prior = 10,
                               alpha_phi_prior = 10,
                               ##tp = tp,
                               ## max threshold value.
                               ## having it too close to xmax *sometimes* leads to sampling
                               ## near xmax, but with lower lp and very high E13) b0 values
                               y_xmax = 0),
                   cores = n_cores,
                   chains = n_chains,
                   iter = iter,
                   warmup = floor(iter/2),
                   verbose = FALSE)

        fit_tbl[ curr_row, ]$fit <- list(fit)

    }
}
```

```
## [1] "nb: separate, separate"
## [1] "nb: separate, grouping_1"
## [1] "nb: separate, pooled"
## [1] "nb: grouping_1, separate"
## [1] "nb: grouping_1, grouping_1"
## [1] "nb: grouping_1, pooled"
## [1] "nb: pooled, separate"
## [1] "nb: pooled, grouping_1"
## [1] "nb: pooled, pooled"
```

```r
nb_fit_tbl <- fit_tbl

## save(file = "fit_tbl.Rda", fit_tbl)
```

Model Comparison

LOO Analysis

```r
fit_tbl <- bind_rows(qpoisson_fit_tbl, nb_fit_tbl, .id = NULL)

for(curr_row in 1:length(fit_tbl$fit)) {

  desc <- fit_tbl[[curr_row, "desc"]]
  fit <- fit_tbl[[curr_row, "fit"]][[1]]
  print(paste0("Model ", curr_row, ": ", desc))

  # loo analysis based on: http://mc-stan.org/loo/articles/loo2-with-rstan.html
  #
  # Extract pointwise log-likelihood
  # using merge_chains=FALSE returns an array, which is easier to
  # use with relative_eff()
  llik <- extract_log_lik(fit, merge_chains = FALSE)
  fit_tbl[[curr_row, "llik"]] <- list(llik)

  # as of loo v2.0.0 we can optionally provide relative effective sample sizes
  # when calling loo, which allows for better estimates of the PSIS effective
  # sample sizes and Monte Carlo error
  r_eff <- relative_eff(exp(llik), cores = n_cores)
  fit_tbl[[curr_row, "r_eff"]] <- list(r_eff)

  # preferably use more than 2 cores (as many cores as possible)
  # will use value of 'mc.cores' option if cores is not specified
  loo <- loo(llik, r_eff = r_eff,
             cores = n_cores,
             save_psis = TRUE,
             moment_match = TRUE)
  fit_tbl[[curr_row, "loo"]] <- list(loo)
  print(loo)

}
```

```
## [1] "Model 1: qpoi: grouping_1, grouping_1"
##
## Computed from 30000 by 38 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -276.8 34.9
## p_loo        21.6  7.6
## looic       553.5 69.8
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     33    86.8%   2810
##  (0.5, 0.7]   (ok)        3     7.9%   292
##    (0.7, 1]   (bad)       2     5.3%   22
##    (1, Inf)   (very bad)  0     0.0%   <NA>
## See help('pareto-k-diagnostic') for details.
## [1] "Model 2: qpoi: grouping_1, pooled"
##
## Computed from 30000 by 38 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -298.0 34.1
## p_loo        16.3  5.2
## looic       595.9 68.2
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     35    92.1%   1109
##  (0.5, 0.7]   (ok)        1     2.6%   11233
##    (0.7, 1]   (bad)       1     2.6%   113
##    (1, Inf)   (very bad)  1     2.6%   22
## See help('pareto-k-diagnostic') for details.
## [1] "Model 3: qpoi: grouping_1, separate"
##
## Computed from 30000 by 38 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -291.6 48.7
## p_loo        64.4 24.9
## looic       583.1 97.3
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     22    57.9%   3189
##  (0.5, 0.7]   (ok)        8    21.1%   904
##    (0.7, 1]   (bad)       3     7.9%   35
##    (1, Inf)   (very bad)  5    13.2%   3
## See help('pareto-k-diagnostic') for details.
```

```
## [1] "Model 4: qpoi: pooled, grouping_1"
##
## Computed from 30000 by 38 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -283.7 36.8
## p_loo        28.0  9.6
## looic       567.4 73.6
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     33    86.8%   274
##  (0.5, 0.7]   (ok)        3     7.9%   160
##    (0.7, 1]   (bad)       1     2.6%   26
##    (1, Inf)   (very bad)  1     2.6%   59
## See help('pareto-k-diagnostic') for details.
## [1] "Model 5: qpoi: pooled, pooled"
##
## Computed from 30000 by 38 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -305.5 36.0
## p_loo        20.3  6.4
## looic       611.0 72.0
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     36    94.7%   717
##  (0.5, 0.7]   (ok)        1     2.6%   469
##    (0.7, 1]   (bad)       1     2.6%   30
##    (1, Inf)   (very bad)  0     0.0%   <NA>
## See help('pareto-k-diagnostic') for details.
## [1] "Model 6: qpoi: pooled, separate"
##
## Computed from 30000 by 38 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -298.9 45.9
## p_loo        63.0 22.8
## looic       597.8 91.9
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     25    65.8%   323
##  (0.5, 0.7]   (ok)        6    15.8%   48
##    (0.7, 1]   (bad)       4    10.5%   19
##    (1, Inf)   (very bad)  3     7.9%   1
## See help('pareto-k-diagnostic') for details.
```

```
## [1] "Model 7: qpoi: separate, grouping_1"
##
## Computed from 30000 by 38 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -285.1 40.4
## p_loo        42.1 15.7
## looic       570.2 80.7
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     31    81.6%   1885
##  (0.5, 0.7]   (ok)        3     7.9%   388
##    (0.7, 1]   (bad)       2     5.3%   52
##    (1, Inf)   (very bad)  2     5.3%   13
## See help('pareto-k-diagnostic') for details.
## [1] "Model 8: qpoi: separate, pooled"
##
## Computed from 30000 by 38 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -302.0 39.0
## p_loo        32.4 10.2
## looic       604.1 78.0
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     23    60.5%   1146
##  (0.5, 0.7]   (ok)        5    13.2%   3817
##    (0.7, 1]   (bad)       5    13.2%   67
##    (1, Inf)   (very bad)  5    13.2%   2
## See help('pareto-k-diagnostic') for details.
## [1] "Model 9: qpoi: separate, separate"
##
## Computed from 30000 by 38 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -297.4 49.7
## p_loo        69.1 25.2
## looic       594.8 99.4
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     20    52.6%   1626
##  (0.5, 0.7]   (ok)       10    26.3%   346
##    (0.7, 1]   (bad)       4    10.5%   32
##    (1, Inf)   (very bad)  4    10.5%   4
## See help('pareto-k-diagnostic') for details.
```

```
## [1] "Model 10: nb: grouping_1, grouping_1"
##
## Computed from 30000 by 38 log-likelihood matrix
##
##          Estimate  SE
## elpd_loo   -215.3 4.9
## p_loo         2.0 0.9
## looic       430.6 9.7
## ------
## Monte Carlo SE of elpd_loo is 0.0.
##
## All Pareto k estimates are good (k < 0.5).
## See help('pareto-k-diagnostic') for details.
## [1] "Model 11: nb: grouping_1, pooled"
##
## Computed from 30000 by 38 log-likelihood matrix
##
##          Estimate  SE
## elpd_loo   -219.0 4.7
## p_loo         1.7 0.5
## looic       438.1 9.5
## ------
## Monte Carlo SE of elpd_loo is 0.0.
##
## All Pareto k estimates are good (k < 0.5).
## See help('pareto-k-diagnostic') for details.
## [1] "Model 12: nb: grouping_1, separate"
##
## Computed from 30000 by 38 log-likelihood matrix
##
##          Estimate  SE
## elpd_loo   -219.1 4.0
## p_loo         4.3 1.1
## looic       438.2 8.0
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     33    86.8%   7133
##  (0.5, 0.7]   (ok)        4    10.5%   2175
##    (0.7, 1]   (bad)       1     2.6%   2960
##    (1, Inf)   (very bad)  0     0.0%   <NA>
## See help('pareto-k-diagnostic') for details.
## [1] "Model 13: nb: pooled, grouping_1"
##
## Computed from 30000 by 38 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -215.5  5.1
## p_loo         1.9  0.9
## looic       430.9 10.1
## ------
## Monte Carlo SE of elpd_loo is 0.0.
```

```
##
## All Pareto k estimates are good (k < 0.5).
## See help('pareto-k-diagnostic') for details.
## [1] "Model 14: nb: pooled, pooled"
##
## Computed from 30000 by 38 log-likelihood matrix
##
##          Estimate  SE
## elpd_loo   -218.7 4.7
## p_loo         1.3 0.4
## looic       437.4 9.4
## ------
## Monte Carlo SE of elpd_loo is 0.0.
##
## All Pareto k estimates are good (k < 0.5).
## See help('pareto-k-diagnostic') for details.
## [1] "Model 15: nb: pooled, separate"
##
## Computed from 30000 by 38 log-likelihood matrix
##
##          Estimate  SE
## elpd_loo   -219.7 4.2
## p_loo         4.5 1.2
## looic       439.5 8.3
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     34    89.5%   7574
##  (0.5, 0.7]   (ok)        2     5.3%   3068
##    (0.7, 1]   (bad)       2     5.3%   581
##    (1, Inf)   (very bad)  0     0.0%   <NA>
## See help('pareto-k-diagnostic') for details.
## [1] "Model 16: nb: separate, grouping_1"
##
## Computed from 30000 by 38 log-likelihood matrix
##
##          Estimate  SE
## elpd_loo   -215.6 4.8
## p_loo         2.3 1.0
## looic       431.1 9.6
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     37    97.4%   17849
##  (0.5, 0.7]   (ok)        0     0.0%   <NA>
##    (0.7, 1]   (bad)       1     2.6%   915
##    (1, Inf)   (very bad)  0     0.0%   <NA>
## See help('pareto-k-diagnostic') for details.
## [1] "Model 17: nb: separate, pooled"
##
```

```
## Computed from 30000 by 38 log-likelihood matrix
##
##          Estimate  SE
## elpd_loo   -218.7 4.6
## p_loo         1.9 0.6
## looic       437.3 9.2
## ------
## Monte Carlo SE of elpd_loo is 0.0.
##
## All Pareto k estimates are good (k < 0.5).
## See help('pareto-k-diagnostic') for details.
## [1] "Model 18: nb: separate, separate"
##
## Computed from 30000 by 38 log-likelihood matrix
##
##          Estimate  SE
## elpd_loo   -219.4 3.9
## p_loo         4.3 1.1
## looic       438.9 7.8
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     33    86.8%   8535
##  (0.5, 0.7]   (ok)        4    10.5%   3318
##    (0.7, 1]   (bad)       1     2.6%   2695
##    (1, Inf)   (very bad)  0     0.0%   <NA>
## See help('pareto-k-diagnostic') for details.
```

```
comp <- loo_compare(fit_tbl$loo)
index <- comp %>% rownames() %>% sub(pattern = "model", x= ., "") %>% as.integer()
desc <-fit_tbl$desc[index]
rownames(comp) <- desc

#loo_tbl<- bind_cols( desc = desc, index = index, comp) %>% tibble() %>% arrange(index)

print(comp)
```

```
##                                elpd_diff se_diff
## nb: grouping_1, grouping_1     0.0       0.0
## nb: pooled, grouping_1        -0.2       0.4
## nb: separate, grouping_1      -0.3       0.2
## nb: separate, pooled          -3.4       1.6
## nb: pooled, pooled            -3.4       1.7
## nb: grouping_1, pooled        -3.7       1.6
## nb: grouping_1, separate      -3.8       1.2
## nb: separate, separate        -4.1       1.3
## nb: pooled, separate          -4.4       1.0
## qpoi: grouping_1, grouping_1 -61.4      33.5
## qpoi: pooled, grouping_1     -68.4      35.1
## qpoi: separate, grouping_1   -69.8      38.8
## qpoi: grouping_1, separate   -76.3      47.8
## qpoi: separate, separate     -82.1      48.7
```

```
## qpoi: grouping_1, pooled     -82.7      32.2
## qpoi: pooled, separate       -83.6      44.8
## qpoi: separate, pooled       -86.7      37.6
## qpoi: pooled, pooled         -90.2      34.4
```

Plot Results

```
for(curr_row in 1:length(fit_tbl$fit)) {

    desc <- fit_tbl[[curr_row, "desc"]]
    fit <- fit_tbl[[curr_row, "fit"]]

    x0_group_list <- fit_tbl[[curr_row, "x0_group_list"]][[1]]
    y0_group_list <- fit_tbl[[curr_row, "y0_group_list"]][[1]]


}



    pars <- c("x0", "y0", "theta")
    pars_full <- c(pars, "lp__")

    print(desc)
    print(fit, pars = pars)
                                        #traceplot(model, pars = pars, inc_warmup = FALSE)
                                        #plot(model, pars = pars) #, ggtitle(title))

        pairs(model, pars = pars_full)

        ## Plot parameter estimate summaries
        tmp_plot <- list()
        for(par in pars) {
            tmp_plot[[par]] <- stan_plot(model, pars = par)
        }
        gt <- arrangeGrob(grobs = tmp_plot)
        as_ggplot(gt)

    }
}
```

```
## Error: <text>:32:5: unexpected '}'
## 31:
## 32:     }
##         ^
```