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

Michael Gilchrist

Created: 2023-05-18; Compiled: Fri May 19 11:19:53 2023

History

- 2023-05-18: COpied files from `2023-03-18_fit.real.data.using.nbinom_type1-vary.disp`

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

- Earlier work generated poor estimates of `x0`.
- Visualization of data and model fit indicates there's very little information on x0.
- While I can generate predictions of expected values, I can't generate expected values of the data itself. I expect this is due to fact that we generate parameters which result in y = `NaN`
- TODO
- Figure out better model definition that avoids generating NaN values. I expect this can be done by imposing a better prior on `x0|male`.
- Allow disp to vary between males.
- On 3/22/2023 I added the missing $|dg^{-1}(disp)/ddisp| = |-mu/disp^2| = mu/disp^2$ term to llikelihood function
- On 05/18/2023 I
  - removed the 'missing term' from the llikelihood function given that it seems unnecessary and prevented test regression for `custom_family/nbinom_typ1` from working properly

Insights

- When the `disp` (dispersal or `theta`) gets unrealistically large, we get the emergence of a bimodal 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
- Two males have fitting issues, "T235" and "T236". This appears to be due to a bimodal posterior surface where one region has low 'x0' (< 40C), but low 'y0', ane the other has a high `x0` and low `y0`

Set up

Install libraries

```
## load libraries
library(MASS) # provides negative binomial fitting:  glm.nb
library(stats)
library(tidyverse)
library(brms)
library(loo)
library(ggplot2)
#library(tidybayes)
library(ggpubr)
library(grid)
library(gridExtra)
library(ragg)
library(GGally)
library(cowplot)
library(bayesplot)
ggplot2::theme_set(theme_default(base_size = 10))
#ggplot2::theme_set(theme_default(plot.background = element_rect(color = "black")))

library(broom)
library(viridisLite)
library(cmdstanr)
library(rstan)
options(mc.cores = (parallel::detectCores()-2))
rstan_options(auto_write = TRUE)

## 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 Files

Local Functions

```
source("../Local.Functions/local.functions_ZFI.fittings.R")


which_switch_male <- function(flag) {
  return <- NA #default value
```

```
  if(flag %in% c("uniform_1", "groups_1")) return <- "single"
  if(flag %in% c("uniform_2", "groups_2")) return <- "double"
  if(flag %in% c("individual")) return <- "individual"

  return(return)
}
```

Custom family

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

Load Data

```
infiles <- file.path("input", dir("input"))
print(infiles)
```

```
## [1] "input/data_ind.Rda"
## [2] "input/data.processing_2022-12-15.Rda"
## [3] "input/obs_summary_stats.Rda"
## [4] "input/stats_ind.Rda"
```

```
sapply(infiles,
       load, verbose = TRUE, envir = .GlobalEnv)
```

```
## Loading objects:
##   data_ind
## Loading objects:
##   motif_data
##   motif_data_40C
##   motif_stats
##   motif_stats_40C
##   bird_bill_data
## Loading objects:
##   summary_stats
## Loading objects:
##   stats_ind
```

```
## $`input/data_ind.Rda`
## [1] "data_ind"
##
## $`input/data.processing_2022-12-15.Rda`
## [1] "motif_data"      "motif_data_40C"  "motif_stats"     "motif_stats_40C"
## [5] "bird_bill_data"
##
## $`input/obs_summary_stats.Rda`
## [1] "summary_stats"
##
## $`input/stats_ind.Rda`
## [1] "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_count temp_target  temp round trial_round date     counter
##   <chr> <int>       <int>       <dbl> <dbl> <dbl>       <dbl> <chr>    <chr>
## 1 T234      1           0          42  43.0     1           1 02/03/22 RAS
## 2 T234      1          30          44  44.5     1           2 02/05/22 RAS
## 3 T234      1          34          27  27.2     1           3 02/07/22 RAS
## 4 T234      1          87          40  41.1     1           4 02/09/22 RAS
## 5 T234      1          52          35  36.1     1           5 02/11/22 RAS
## 6 T234      1          32          40  39.5     2           1 04/23/22 KIM
## # i 2 more variables: y0_simple_est <dbl>, phi_ind <dbl>
```

```
names(data_ind)
```

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

Local Settings

```
#set seom variables that I expect to change below

display_plots <- FALSE
save_plots_file <- FALSE
color_scheme_set("viridis")
```

Determine reasonable priors for y0

```
cumulative_count_vs_temp <- list()

for(temp_threshold in 26:45) {
  tmp <- data_ind %>% group_by(male) %>% filter(temp < temp_threshold) %>% summarize(mean = mean(motif_
  mean <- mean(tmp$mean)
  sd <- sd(tmp$mean)
  cumulative_count_vs_temp[[temp_threshold]] <- tmp
  print(paste0("Temp: ", temp_threshold, ", mean: ", mean, ", sd: ", sd))
}

plot_pairs <- pairs(cumulative_count_vs_temp[[39]] %>% select(-c(male, sd, cv)))

if(display_plots) print(last_plot())
```

```
hist <- ggplot(cumulative_count_vs_temp[[39]], aes(mean)) +
  geom_histogram(bins = 6)

hist_log <- hist + scale_x_log10()

plot_grid <- plot_grid(plotlist = list(hist, hist_log))
```

If using a normal prior, go with `mean = 125` and `sd = 125 * 4 = 500`. However, the data doesn't seem to follow any real distribution and its at the motif rather than song scale, as a result each male has its own, unknown scaling factor between motifs and songs (i.e. 1/E(# motifs/song) so a flat prior is justifiable.

Fit Models

- Code derived from `../2023-02-28_fit.real.data.using.nbinom_type1/brms_two.piece_fit.nbionm_type1.Rmd`

Set up functions, parameters, and results tibble

```
data_tbl <- data_ind %>%
  rename(y = motif_count, x = temp) #%>%
#mutate(male = factor(male))
males <- unique(data_tbl$male)
nmales <- length(males)

xmax <- 46 # maximum value for x0
xignore <- 39 # x value above which data is ignored in one_piece model

stan_two_piece_func <- paste0(" real  two_piece(real x, real x0, real y0) {
 real xmax = ", xmax, "; ## paste in value for xmax
 real y;

 if(x0 > xmax) {
    y = log(0);
 } else {
    y = y0 * (xmax - fmax(x0, x))/(xmax - x0);
 }
 return(y);
 }\n")

stan_one_piece_func <- paste0(" real  one_piece(real y0) {
 return(y0);
 }\n")

stan_asymptotic_func <- paste0(" real  asymp(real x, real phi, real y0) {
 real xmax = ", xmax, "; ## paste in value for xmax\n
 return(y0 * (1 - exp( - phi * (xmax - x))) );
 }\n")

cat(stan_two_piece_func)
```

```
##  real  two_piece(real x, real x0, real y0) {
##  real xmax = 46; ## paste in value for xmax
##  real y;
##
```

```
##  if(x0 > xmax) {
##     y = log(0);
##  } else {
##     y = y0 * (xmax - fmax(x0, x))/(xmax - x0);
##  }
##  return(y);
##  }
```

Set up Dataframe for fit results

- In retrospect, I should just define the columns and then populate the cells when I do my fittings.

```
fit_tbl_initiate_crossed <- TRUE

models <- c("one_piece", "two_piece", "asymptotic") #, "one_piece") #, "asymptotic")
sampling_dists <- c("nbinom_type1") ##, "com_poisson") ## lognormal doesn't work since the counts can b
flags_x0 <- c("uniform_1",
              "uniform_2",
              "groups_1", ## this doesn't work with x0_Intercept prior, suggests error in priors
              "groups_2", #               "groups_2a",
              #               "groups_2b",
              "individual")

flags_y0 <- c("uniform_1", "groups_1", "individual") %>% last()

values_disp <- switch(1,
                      c(0.01), # 0.125 is a good value
                      c(0.01, 0.1), #, 0.25), # used in exploring model behavior
                      list(0.1, "flat"), #, 0.1, 1) # doesn't work yet
                      c("flat"))

flags_disp <- c("uniform_1", "uniform_2", "groups_1", "individual")


## whether to filter males with large disp values estimated using one piece model
filter_male <- c(TRUE, FALSE)

N <- length(data)

## define full table a priori
fit_tbl_crossed <- crossing(
  model = models,
  #sampling_dist = sampling_dists,
  x0_flag = flags_x0,
  y0_flag = flags_y0,
  disp_flag = flags_disp,
  disp_value = values_disp,
  filter_male = filter_male,
  fit = list(NA),
  stats_fit = list(NA),
  fit_best = list(NA),
  fit_top = list(NA),
  plots = list(list()),
  llik = list(NA),
  r_eff = list(NA),
```

```
  loo = list(NA)
)


if(fit_tbl_initiate_crossed) {

  fit_tbl <- fit_tbl_crossed
} else {

  ## Use an empty tibble
  fit_tbl <- fit_tbl_crossed[0, ]
}
```

Run fit

Note

- Code starting wth `results=...` added to allow use of `\clearpage` For more details, see: [https://stackoverflow.com/a/48069427/5322644].

```
##, results='asis', eval=(knitr::opts_knit$get('rmarkdown.pandoc.to') == 'latex'), echo = TRUE}

if(interactive()) {
    ## Run Settings
    run_fits <- TRUE
    ## run_fits <- FALSE
    save_fits <- FALSE
    save_fits <- TRUE
    run_fits_force <- TRUE # force refitting of model; used with run_fits = TRUE
    load_fits <- FALSE  # reload fit_tbl even if it already exits; used with run_fits = FALSE
    verbose <- TRUE

    ## Fit Settings
    render_plots <- TRUE ## Build or use previously generated plots in tbl
    render_plots_force <- TRUE ## Force regeneration of plots if they already exist
    render_hex <- TRUE
    render_hist <- TRUE
    render_pairs <- FALSE
    off_diag_fun = "hex"
    display_plots <- FALSE ## Display plots to local device?
    save_plots_file <- TRUE  ## Save plots to file?
} else {

    ## Non-interactive (Knitr) settings
    run_fits <- FALSE
    save_fits <- FALSE
    ## save_fits <- TRUE
    run_fits_force <- FALSE # force refitting of model; used with run_fits = TRUE
    load_fits <- TRUE  # load fit_tbl below?
    verbose <- TRUE

    ## Fit Settings
    render_plots <- TRUE ## Build or use previously generated plots in tbl
    render_plots_force <- TRUE ## Force regeneration of plots if they already exist
    render_hex <- TRUE
```

```
    render_hist <- TRUE
    render_pairs <- TRUE
    off_diag_fun <- "hex"
    display_plots <- TRUE
    save_plots_file <- TRUE
}


## Print settings
print_get_prior <- TRUE ## Print generic priors, prior to fit.
print_prior_summary <- TRUE  ## Print actual priors, post fitting

if(run_fits_force) {
  infile_tbl <- NULL
} else {
  infile <- last(dir(file.path(output_dir, "tibbles"), "fit_tbl.*"))
  ## over ride latest file below
  #infile <- "fit_tbl_adapt-delta-0.90.Rda"
  infile_tbl <- file.path(output_dir, "tibbles", infile)

  if(load_fits) {
    print(paste0("Loading `infile_tbl` = ", infile_tbl))
    load(infile_tbl)
    #fit_tbl[["plots"]] <- list()
  }

}
```

```
## [1] "Loading `infile_tbl` = output/render/tibbles/fit_tbl_2023-05-18_14.47.35.343213.Rda"
```

```
if(save_fits) {
  cur_time <- gsub(" ", "_", Sys.time()) %>% gsub(":", ".", .)
  outfile_tbl <- file.path(output_dir, "tibbles", paste0("fit_tbl_", cur_time, ".Rda"))
} else {
  outfile_tbl <- NA ## NULL causes an error when trying to print
}

print_pairs_plot <- FALSE  # Could base on model used, gets large when lots of individual or groups

sampling = "nbinom_type1"
prior_shape_y0 = "flat"

n_chains <- 10
n_cores <- n_chains
n_chains_top <- 4  # how many to keep for model analysis

flags_x0_used <- c("individual", "groups_1", "uniform_1") # %>% rev()#
flags_y0_used <- c("individual")
values_disp_used <- values_disp
flags_disp_used <- c("individual", "groups_1", "groups_2", "uniform_1", "uniform_2")[4:5] #  |> rev() |
models_used <- c("one_piece", "two_piece", "asymptotic")[2] #c("one_piece", "two_piece") #, "two_piece"
shape_y0_prior <- "flat" # flat or normal

## These males produce bimodial posteriors and interfere with model fitting
```

```r
## Ideally, we'd do a preliminary analysis without them and then include them later.
#male_filter = c("T235", "T236")

## These males have huge dispersion outside the predicted valeus
male_disp_high <- c("T235", "T243")
male_x0_high <- NA ## Define later
male_filter_out <- c("T235", "T243")

curr_row <- 1


for(model in models_used) {
  print(model)
  switch(model,
         two_piece = {
           ## Note issues in non-convergence are related to bimodality of posterior surface.
           stan_func <- stan_two_piece_func
           warmup <- 30000 # floor(3/4 * iter)
           iter <- warmup + 10000
           adapt_delta <- 0.9 ## increasing from 0.9 to 0.99 didn't help eliminate divergence
           thin <- 5
         },
         one_piece = {
           stan_func <- stan_one_piece_func
           warmup <- 2000
           iter <- warmup + 2000
           thin <- 4
           adapt_delta <- 0.7
         },
         asymptotic = {
           stan_func <- stan_asymptotic_func
           warmup <- 2000
           iter <- warmup + 2000
           thin <- 4
           adapt_delta <- 0.9
         }

         )

  cat(stan_func)


  for(male_filter in c(FALSE)) { #, TRUE)) {

    for(disp_flag in flags_disp_used) {

      for(x0_flag in flags_x0_used) {

        for(y0_flag in flags_y0_used) {

          ## define variable for labeling figures
          x0_label <- ifelse(model == "one_piece", "NA", x0_flag)
```

```
for(disp_value in values_disp_used) {

  ## Set up variables for saving model and fit

  desc_short <- paste0("x0: " , x0_label, "; y0: ", y0_flag, "; disp_flag: ", disp_flag, "; di
  desc <- paste0(sampling, "; ", model, "; ", desc_short)
  #print(desc)

  desc_filename <- gsub("_|\\.", "-", desc) %>%
    gsub("; ", "_", .) %>%
    gsub(":? ", "-", .)

  #stan_model_name <- sub(desc_filename, "_disp-prior-[0-9.]+_filter", "_filter")
  stan_model_name <- desc_filename #sub("_disp-prior-[0-9.]+_", "_", desc_filename)


  if(run_fits) {

    ifelse(length(fit_tbl) >= curr_row, fit_prerun <- fit_tbl[[curr_row, "fit"]][[1]], fit_pre
    if(run_fits_force | !is_brmsfit(fit_prerun)) {
      print("Fitting Models")
      switch(sampling,
             "nbinom_type1"= {
               family <- nbinom_type1(link = "identity", link_disp = "identity")
               adapt_delta <- adapt_delta #0.95 ## will decreasing value increase ESS? Seems
               iter <- iter
               warmup <- warmup
               thin <- thin
               n_cores <- n_cores ## set to 1 if getting errors from stan in order to see rel
               n_chains <- n_chains
               stanvar_func <-
                 stanvar(scode = paste(
                   stan_func,
                   stan_nbinom_type1, sep = "\n"),
                   block = "functions")
             }
             )

      ## Refresh data in case x0_group or y0_group are all set to 1
      data <- data_tbl

      if(model == "one_piece") data <- data %>% filter(x < xignore)
      if(male_filter) data <- data %>%
                        filter(!(male %in% male_filter_out))

      print("Set grouping flags and formula")

      for(flag_prefix in c("x0", "y0", "disp")) {

        group_txt <- paste0(flag_prefix, "_group")
        flag_txt <- paste0(flag_prefix, "_flag")
        flag <- eval(parse(text = flag_txt))
```

```
  ## Map from flags to more general categories below: "single", "double"...
  switch_male <- which_switch_male(flag)

  data <- switch(switch_male,
                 "single" = {
                   mutate(data, tmp_group = 1)
                 },
                 "double" = {
                   male_high <- eval(parse(text=paste0("male_", flag_prefix, "_high")))
                   mutate(
                     data,
                     tmp_group =
                       as.factor(if_else(male %in% male_high, 2, 1)))
                 },
                 "individual" = {
                   mutate(data, tmp_group = 1) #tmp_group = male)
                 },
                 NA
                 ) %>%
    rename(!!group_txt := tmp_group)


  ## Define model formulas
  flag_txt <- paste0(flag_prefix, "_flag")
  flag <- eval(parse(text = flag_txt))

  print(paste0(flag_prefix, ": ", flag))

  flag_str <- paste0(flag_prefix, "_group")

  string_formula <- switch(flag,
                           uniform_1 = paste0(flag_prefix, " ~ 0 + Intercept"),
                           ## Doesn't work: paste0(flag_prefix, " ~ 0 + ", flag_str),
                           uniform_2 = paste0(flag_prefix, " ~ 0 + ", flag_str),
                           ## `0 + Intercept` avoids prior being defined on centered da
                           ## The following formulation is incorrect and
                           ## assumes that x0 is centered around 0
                           ## groups_1 = formula(x0 ~ 0 + (1|male)),
                           groups_1 = paste0(flag_prefix, " ~ 0 + Intercept + (1|male)"
                             ##paste0(flag_prefix, " ~ 0 + ", flag_str, " + (", flag_st


                           groups_2 = paste0(flag_prefix, " ~ 0 + Intercept + (1|male) +
                           ##paste0(flag_prefix, " ~ 0 + (", flag_str, "|male)"),

                           individual = paste0(flag_prefix, " ~ 0 + male"), ## Do not u
                           NA
                           )
  form_txt <- paste0(flag_prefix, "_form")
  assign(form_txt, as.formula(string_formula))
}

## str_extract_all(names(data), ".*_flag") %>%
```

```
##   unlist() %>%
##   print()

## Used in asymptotic model: use phi instead of threshold x0
phi_form <- formula(deparse(x0_form) %>% gsub("x0", "phi", .))

threshold_form <- switch(model,
                         two_piece = x0_form,
                         one_piece = NULL,
                         asymptotic = phi_form
                         )

nlform <- switch(model,
                 two_piece = bf(y ~ two_piece(x, x0, y0), nl = TRUE),
                 one_piece = bf(y ~ one_piece(y0), nl = TRUE),
                 asymptotic = bf(y ~ asymp(x, phi, y0), nl = TRUE)
                 ) +
  threshold_form +
  disp_form +
  y0_form


print("Define priors")

## pass disp_value via stanvar argument
stanvar_prior <- stanvar(disp_value, name = "disp_value")

prior_string <- if(disp_value == "flat") {
  "uniform(0, 20)"
} else {
  # encode non-flat prior here, which force recompling when disp_value changes
  #paste0("exponential(", disp_value, ")")
  # pass disp_value via stanvar argument
  # Allows disp_value to be changed w/o recompiling
  "exponential(disp_value)"
}
disp_priors <- switch(disp_flag,
                      #uniform_1 = set_prior(prior_string, class = "disp", lb = 0, ub =
                      ## Form when disp ~ 1:
                      uniform_1 = set_prior(prior_string, class = "b", dpar = "disp", l
                      uniform_2 = NULL, ## This is probably broken
                      groups_1 = set_prior(prior_string,
                                           class = "b", dpar = "disp", lb = 0, ub = 20)
                        set_prior("uniform(0.1, 5)", class = "sd", dpar = "disp", lb = 
                      groups_2a = NULL,
                      groups_2b = NULL,
                      individual = set_prior(prior_string,
                                             dpar = "disp", lb = 0, ub = 20)

                      )

## x0 only used in two_piece model
x0_prior <- switch(x0_flag,
```

```
                        uniform_1 = NULL,
                        uniform_2 = NULL,
                        groups_1 = prior(student_t(3, 0, 66.7), lb = 0, ub = 10, class = "sd
                        groups_2 = NULL,
                        individual = NULL
                        )

x0_priors <- prior(uniform(32, 45.9), lb = 32, ub = 45.9, nlpar = "x0") +
  #prior(uniform(32, 45.9), lb = 32, ub = 45.9, nlpar = "x0") +
  x0_prior

phi_priors <- prior(uniform(0.1, 100), lb = 0.01, ub = 17, nlpar = "phi")

y0_priors <- switch(prior_shape_y0,
                    ## Values based on calculations at top of file using `temp_threshol
                    normal = prior(normal(125, 500), nlpar = "y0", lb = 10, ub = 1000),
                    # flat prior
                    # - consistent with fact we're working with motifs, not songs
                    # - avoids bimodal posterior sampling issues with T235 and 236
                    flat =  prior(uniform(10, 1000), nlpar = "y0", lb = 10, ub = 1000)

                    )

threshold_priors <- switch(model,
                           two_piece = x0_priors,
                           one_piece = NULL,
                           asymptotic = phi_priors
                           )

prior <- switch(model,
                one_piece = {
                  y0_priors + disp_priors
                },
                threshold_priors + y0_priors + disp_priors
                )

if(print_get_prior) {
  tmp <- get_prior(nlform,
                   data = data,
                   family = family
                   )
  print(tmp,
        max.print = 500)
}

stan_code <- file.path(output_dir,
                       "stan", "code", paste0(stan_model_name, ".stan"))
## make_stancode( .... save_model = stan_code)
fit <- brm(nlform,
           data = data,
           ## `link` refers to the mapping of the expectation of the distribution: log,
           ## 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,
                        stanvar = stanvar_func + stanvar_prior, ## pass prior values here
                        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.
                                       ),
                        ## Ideally save model to avoid need to recompile
                        stan_model_args = list(
                          model_name = file.path(output_dir, "stan", "binary", stan_model_name)
                        ),
                        #sample_prior =  "no", ## note improper priors not sampled
                        ## 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 = stan_code
                        )

          ## We should repeatedly refit model until we get desired number of fit_best
          #print("Prior Summary")
          #print(prior_summary(fit))
          #print("Fit Information")
          #print(desc)
          #print(fit)

          #fit_exp <- expose_functions(fit) , vectorize = TRUE)
          #fit_cr <- add_criterion(fit_exp, c("loo", "waic"))
          fit_tbl[[curr_row, "fit"]] <- list(fit)
          fit_tbl[[curr_row, "x0_flag"]] <- x0_flag
          fit_tbl[[curr_row, "y0_flag"]] <- y0_flag
          fit_tbl[[curr_row, "disp_flag"]] <- disp_flag
          fit_tbl[[curr_row, "disp_value"]] <- disp_value
          ## Print current warnings
          warnings(summary)
          ## Clear warnings()
        } else {
          print("Using pre existing fit")
          fit <- fit_prerun
        }


        ## Extract up to `n_chains_top` chains for fit_best and fit_top

        verify_brmsfit(fit)
        fit_brms <- fit
        fit_stan <- fit_brms$fit
```

```
    stats_fit <- calc_fit_lp_stats(fit_brms)
    print(stats_fit %>% arrange(desc(mean)))
    fit_tbl[[curr_row, "stats_fit"]][[1]] <- stats_fit

    ## Get all fits similar to the best one
    ## DOESN"T WORK CONSISTENTLY
    ## There's an issue with the local.functions code
    ## not working with both stanfit and brmsfit objects
    ## It's very confusing
    make_stan_best <- TRUE

    if(make_stan_best) {
      fit_stan_best <- keep_chains_top(fit_stan, n_chains_max = n_chains_top, verbose = TRUE,
      ## use existing brms object to update
      fit_best <- fit_brms
      fit_best$fit <- fit_stan_best
      ## update tbl
      ##str(fit)
      fit_tbl[[curr_row, "fit_best"]][[1]] <- fit_best
    }

    ## DOESN"T WORK CONSISTENTLY
    ## There's an issue with the local.functions code
    make_stan_top <- TRUE
    if(make_stan_top) {
      ## Get the top fits
      fit_stan_top <- keep_chains_top(fit_stan, n_chains_max = n_chains_top, verbose = FALSE,
      ## Use current fit object to construct new one
      fit_top <- fit_brms
      fit_top$fit <- fit_stan_top
      ## update tbl
      fit_tbl[[curr_row, "fit_top"]][[1]] <- fit_stan_top
    }
    ## end if(run_fits)
  } else {
    print("Working with Pre-existing Fits")

    ## Try to assign from local memory.
    if(!exists("fit_tbl") | nrow(fit_tbl) == 0) {
      print(paste("Loading Models from file", infile_tbl))
      load(file = infile_tbl, verbose = TRUE)
      ## Need only for older fittings
      fit_tbl <- add_column_safely(fit_tbl, "plots")
    }

    fit <- fit_tbl[[curr_row, "fit"]][[1]]
    print(paste0("`class(fit)` = ", class(fit)))
    fit_stan <- fit$fit
    ## These shoudl be brmsfit objects
    fit_top <- fit_tbl[[curr_row, "fit_top"]][[1]]
    fit_best <- fit_tbl[[curr_row, "fit_best"]][[1]]
    stats_fit <- fit_tbl[[curr_row, "stats_fit"]][[1]]
```

```
} ## end loading of previous fits

cat("\n\n\n")

## lp_stats <- calc_fit_lp_stats(fit)
if(print_prior_summary) {
  print(desc)
  print("Prior Information")
  print(prior_summary(fit)) # %>% filter(nlpar!="y0"))
}

print(desc)
print("Fit Information")
print(summary(fit)) #, pars = "x0*"))          %>% filter(nlpar!="y0"))
print(stats_fit)

## Clean up variable names
fit_stan_rename <-
  fit_stan %>%
  clean_var_names()

data <- fit$data
vars_clean <- names(fit_stan_rename) %>% na.omit(.)
male_vec <- unique(data$male) %>% as.character(.)
## get male specific vars (start with "T")
vars_T <- grep("^T[0-9]{3}", vars_clean, value = TRUE)
vars_Intercept <- grep("Intercept", vars_clean, value = TRUE)
vars_non_T <- vars_clean[!(vars_clean %in% c(vars_T, vars_Intercept))]

drop_lprior <- TRUE
if(drop_lprior) vars_non_T <- str_subset(vars_non_T, "lprior", negate = TRUE)
#print(vars_clean)
##  Get plots for current fit settings
plots_row <- fit_tbl[[ curr_row, "plots"]][[1]]

if(verbose) print("Plotting Trace")
## indicate plot name
plot_name <- "plot_trace"
plot_curr <- plots_row[[plot_name]]
if( (is.null(plot_curr)  | render_plots_force) & render_plots) { ## Generate and append plo
  ## Plotting code goes below here
  plot_trace <- mcmc_trace(fit, pars = c("lp__")) +
    ggtitle(desc_short)
  ## Finish plotting code
  plots_row[[plot_name]] <- plot_trace
} else{
  ## update last_plot() settings
  set_last_plot(plot_trace)
}


if(verbose) print("Plotting violin")
## indicate plot name
```

```
plot_name <- "plot_violin"
plot_curr <- plots_row[[plot_name]]
if( (is.null(plot_curr) | render_plots_force) & render_plots) { ## Generate and append plot
  ## Plotting code goes below here
  ## Plot violin plots of posterior values
  ## Based on: https://cran.r-project.org/web/packages/bayesplot/vignettes/plotting-mcmc-dra
  fit_array <- as.array(fit_stan_rename)
  if(FALSE) { ## use violin plots
    plot_tmp <- mcmc_violin(fit_array, pars = c("lp_"), probs = c(0.1, 0.5, 0.9))
  } else { # use histograms
    plot_tmp <- mcmc_hist_by_chain(fit_array, pars = c("lp_")) #, color_chains = TRUE)
  }
  plot_violin <- plot_tmp +
    yaxis_ticks(on = TRUE) +
    yaxis_text(on = TRUE) +
    ggtitle(desc_short)
  ## Finish plotting code
  plots_row[[plot_name]] <- plot_violin
} else{
  ## update last_plot() settings
  set_last_plot(plot_violin)
}


plot_trace_and_violin <- plot_grid(plotlist = list(plot_trace, (plot_violin + ggtitle(NULL))
                                   ncol = 1,
                                   rel_heights=c(0.3, 1))
if(save_plots_file) last_plot_save(file_prefix = "trace-and-violin")
if(display_plots) print(last_plot())

## Count occurence of each male in model fit_brms
## This is used to control plotting
male_instance <- sapply(male_vec, function(x) {sum(str_detect(x, string=vars_clean))})

if(render_hist) {
  if(verbose) print("Plotting hist")
  ## This one can be replace by plot_hex or scatter
  plot_name <- "plot_hist"
  plot_curr <- plots_row[[plot_name]]
  if( (is.null(plot_curr) | render_plots_force) & render_plots) { ## Generate and append pl
    vars_fit <- names(fit_stan_rename) %>% na.omit(.) %>% sort(., decreasing = TRUE)
    ncol <- 4
    ## Plotting code goes below here
    plot_hist <-  stan_hist(fit_stan_rename,
                            pars = vars_fit,
                            bins = 30,
                            ncol = ncol) +
      ggtitle(desc_short) +
      ## Reduce plot label size
      theme(plot.title = element_text(size = rel(1), face = "bold"))
    ## Finish plotting code
    plots_row[[plot_name]] <- plot_hist
  } else{
```

```
    ## update last_plot() settings
    set_last_plot(plot_hist)
  }
  file_prefix <- sub("plot_", "", plot_name)
  if(save_plots_file) last_plot_save(file_prefix = file_prefix)
  if(display_plots) print(last_plot())
}

# remove old scatter plots
# plots_row[["plot_scatter"]] <- NULL
if(render_hex) {
  plot_name <- "plot_hex"
  plot_curr <- plots_row[[plot_name]]
  if( (is.null(plot_curr) | render_plots_force) & render_plots) { ## Generate and append pl
    print("Plotting Hex")
    scatter_list <- list()

    for(male in male_vec) {
      # print(male)
      vars_male <- grep(male, vars_T, value = TRUE)
      #                   mutate(x = gsub("T[0-9]{3}_","", x), y = gsub("T[0-9]{3}_","", y))
      if(length(vars_male) == 1) {
        ## include lp
        vars_male <- c(vars_male, "lp_")
        ylab <- "log(Posterior)"
      } else {

        if(length(vars_male) > 2) {
          print(
            paste0("WARNING: mcmc_hex can only work with 2 variables, but length(vars_male)
                   length(vars_male),
                   "; vars_male = ",
                   paste(vars_male),
                   " Using only first two elements.",
                   collapse = ","))
          vars_male <- vars_male[1:2]
        }

        ylab <- "y0"

      }

      scatter_tmp <-  mcmc_hex( #was mcmc_scatter
        fit_stan_rename,
        ## Can only use two variables
        pars = c(first(vars_male), last(vars_male)),
        ) +
          ##labs(x = "x0", y = ylab, title = male)
          labs(title = male)
      ## Add histograms to plot axes/margin
      ## ggMarginal doesn't work natively with mcmc_hex, so we need to make the
      ## points transparent and then add a hex layer
      scatter_list[[male]] <- ggExtra::ggMarginal(scatter_tmp +
```

```
                                                    geom_point(col="transparent") +
                                                    geom_hex() +
                                                    theme(legend.position = "none"), type =

      }

      p <- plot_grid(plotlist = scatter_list,
                     ncol = 3) #, legend = TRUE)

      title_row <- ggdraw() + draw_label(desc_short, fontface='bold', size = 10)

      plot_hex <- plot_grid(title_row, p, ncol = 1, rel_heights=c(0.1, 1))

      ## Finish plotting code
      plots_row[[plot_name]] <- plot_hex
    } else {
      ## update last_plot() settings
      set_last_plot(plot_hex)
    }

    file_prefix <- sub("plot_", "", plot_name)
    if(save_plots_file) last_plot_save(file_prefix = file_prefix)
    if(display_plots) print(last_plot())
  }

  if(render_pairs) {
    ## plot_Pairs not that useful
    ## Might want to flag off
    plot_name <- "plot_pairs"
    plot_curr <- plots_row[[plot_name]]
    if( (is.null(plot_curr) | render_plots_force) & render_plots) { ## Generate and append pl
      print("Plotting Pairs")
      ## Pairs Plot
      list_plot <- list()
      if(model == "one_piece" & disp_flag != "individual") {
        ## each male only appears once in the one piece models
        splitby <- 3
        nsplits <- max(nmales %/% splitby, 1)
        tmp_stop <- (1:nsplits) * splitby
        tmp_start <- tmp_stop - (splitby - 1)
        for(i in 1:nsplits) {
          tmp_range <- tmp_start[[i]]:tmp_stop[[i]]
          list_plot[[i]] <- pairs(fit,
                                  off_diag_fun = if_else(off_diag_fun == "hex", "hex", "scatt
                                  variable = c(as.character(males[tmp_range]),
                                               "disp"),
                                  regex = TRUE)
        }
        list_plot[[nsplits+1]] <- pairs(fit,
                                        variable = c("disp", "lprior", "lp_")
                                        )
      } else { # not one_piece disp_flag != individual
```

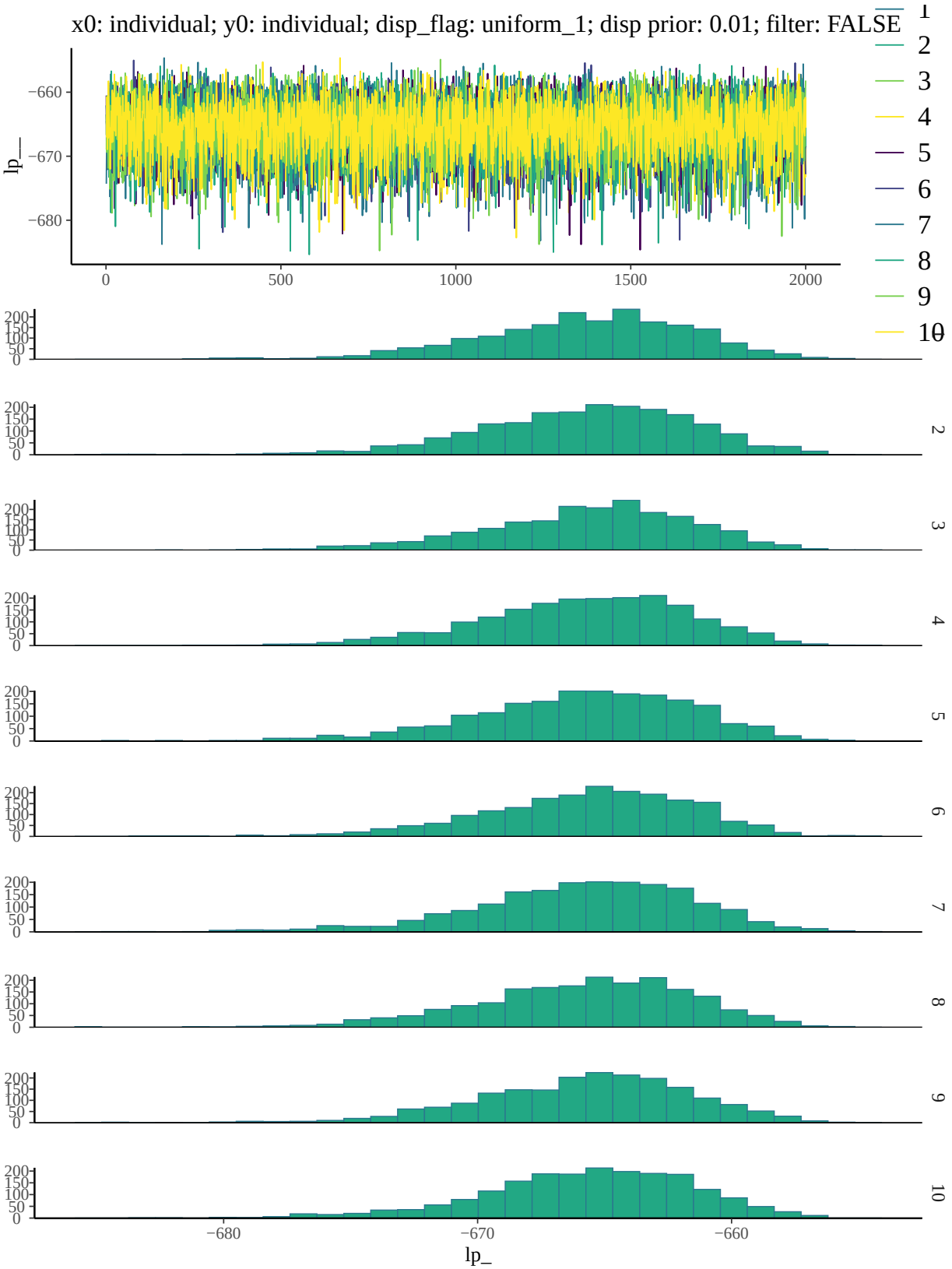
```
            list_plot <- list()
            for(male in male_vec) {
              ##print(male);
              list_plot[[male]] <-
                pairs(fit,
                      off_diag_fun = if_else(off_diag_fun == "hex", "hex", "scatter"),
                      variable = c(male, "lp_"),
                      regex = TRUE)

            }
            #ggtitle(desc_short)
          }

          #browser()

          ## Set number of males per row
          ifelse(length(list_plot[[male]]) > 2, ncol_row <- 1, ncol_row <- 2)
          ## Set number of rows/page
          ifelse(length(list_plot[[male]]) > 2,
                 ifelse(length(list_plot[[male]]) > 4,
                        nrow_page <- 3,
                        nrow_page <- 4),
                 nrow_page <- 5)

          ## Note this makes a list of plots
          plot_pairs <- marrangeGrob(grobs = list_plot,
                                     ncol = ncol_row,
                                     nrow = nrow_page,
                                     top = desc_short,
                                     bottom = quote(paste("page", g, "of", npages))
                                     )
          ## plot_pairs <- cowplot::plot_grid(
          ##   plotlist = list_plot,
          ##   ncol = 2)
          ## Finish plotting code
          #set_last_plot(plot_pairs) ## Don'[t think marrangeGrob updates last_plot()
          plots_row[[plot_name]] <- plot_pairs

        } else {
          ## update last_plot() settings
          set_last_plot(plot_pairs)
        }
        if(display_plots) print(plot_pairs); #dev.off()
        file_prefix <- sub("plot_", "", plot_name)
        ## Need to update following line since imarrangeGrob output isn't technically a plot
        if(save_plots_file) last_plot_save(file_prefix = file_prefix)

        ## update plots_row
        fit_tbl[[ curr_row, "plots"]][[1]] <- plots_row
      }

       #graphics.off()
      # browser()
```

```
            curr_row <- curr_row + 1
            cat("\\clearpage")

          }  ## End disp_values
        } ## end y0_flag
      } ## end x0_flag
    } ## end flag_disp
  } ## end filter male
} ## end model_used
```

```
## [1] "two_piece"
##  real  two_piece(real x, real x0, real y0) {
##  real xmax = 46; ## paste in value for xmax
##  real y;
##
##  if(x0 > xmax) {
##     y = log(0);
##  } else {
##     y = y0 * (xmax - fmax(x0, x))/(xmax - x0);
##  }
##  return(y);
##  }
## [1] "Working with Pre-existing Fits"
## [1] "`class(fit)` = brmsfit"
##
##
##
## [1] "nbinom_type1; two_piece; x0: individual; y0: individual; disp_flag: uniform_1; disp prior: 0.01
## [1] "Prior Information"
##                     prior class      coef group resp dpar nlpar lb   ub
##        uniform(32, 45.9)     b                                x0 32 45.9
##        uniform(32, 45.9)     b  maleT234                    x0 32 45.9
##        uniform(32, 45.9)     b  maleT235                    x0 32 45.9
##        uniform(32, 45.9)     b  maleT236                    x0 32 45.9
##        uniform(32, 45.9)     b  maleT237                    x0 32 45.9
##        uniform(32, 45.9)     b  maleT243                    x0 32 45.9
##        uniform(32, 45.9)     b  maleT244                    x0 32 45.9
##        uniform(32, 45.9)     b  maleT246                    x0 32 45.9
##        uniform(32, 45.9)     b  maleT247                    x0 32 45.9
##        uniform(32, 45.9)     b  maleT257                    x0 32 45.9
##        uniform(32, 45.9)     b  maleT258                    x0 32 45.9
##        uniform(32, 45.9)     b  maleT260                    x0 32 45.9
##        uniform(10, 1000)     b                              y0 10 1000
##        uniform(10, 1000)     b  maleT234                    y0 10 1000
##        uniform(10, 1000)     b  maleT235                    y0 10 1000
##        uniform(10, 1000)     b  maleT236                    y0 10 1000
##        uniform(10, 1000)     b  maleT237                    y0 10 1000
##        uniform(10, 1000)     b  maleT243                    y0 10 1000
##        uniform(10, 1000)     b  maleT244                    y0 10 1000
##        uniform(10, 1000)     b  maleT246                    y0 10 1000
##        uniform(10, 1000)     b  maleT247                    y0 10 1000
##        uniform(10, 1000)     b  maleT257                    y0 10 1000
##        uniform(10, 1000)     b  maleT258                    y0 10 1000
##        uniform(10, 1000)     b  maleT260                    y0 10 1000
```

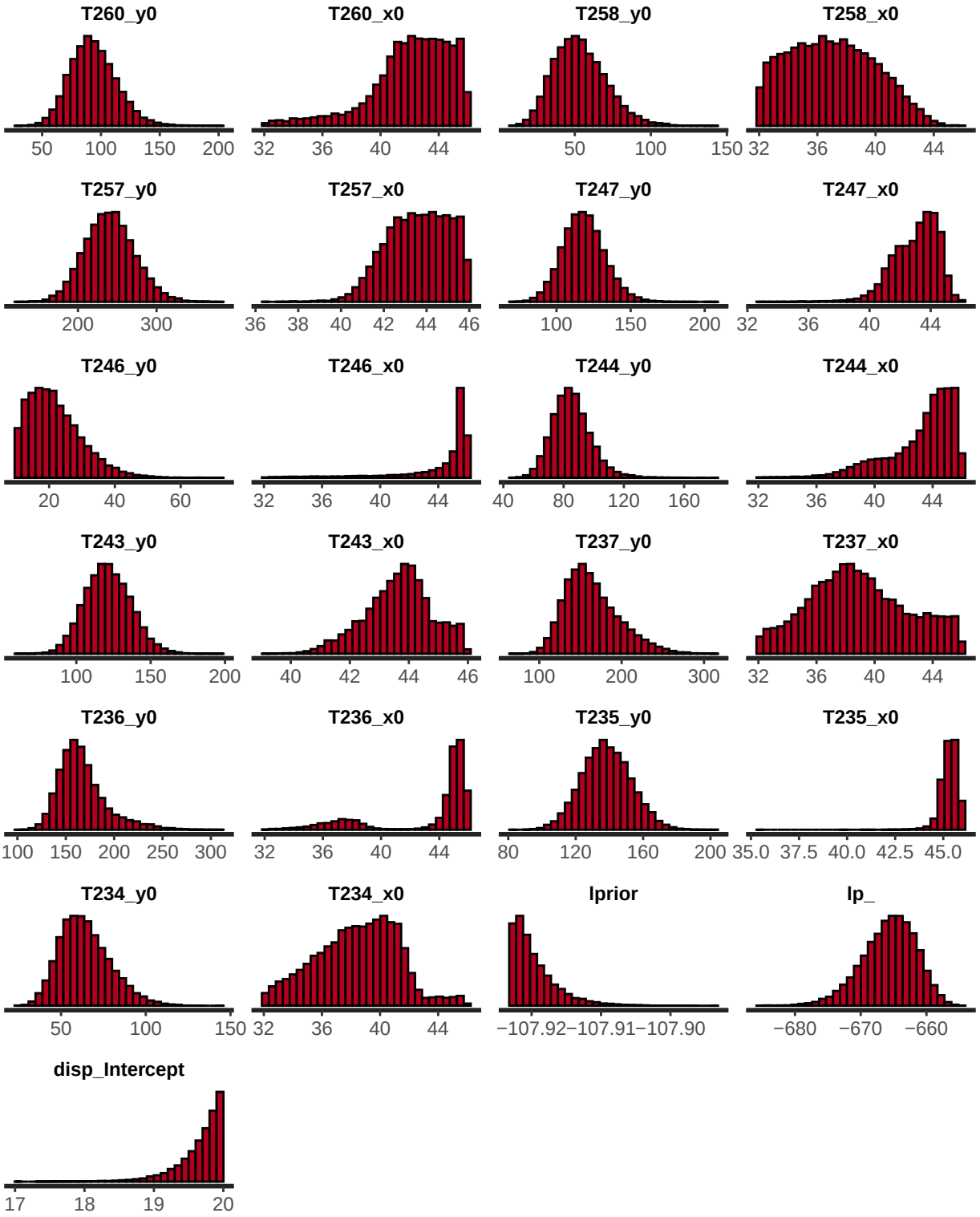
```
##  exponential(disp_value)     b                       disp        0   20
##  exponential(disp_value)     b Intercept             disp        0   20
##        source
##          user
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
##          user
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
##          user
##  (vectorized)
## [1] "nbinom_type1; two_piece; x0: individual; y0: individual; disp_flag: uniform_1; disp prior: 0.01
## [1] "Fit Information"
##  Family: nbinom_type1
##   Links: mu = identity; disp = identity
## Formula: y ~ two_piece(x, x0, y0)
##          x0 ~ 0 + male
##          disp ~ 0 + Intercept
##          y0 ~ 0 + male
##    Data: data (Number of observations: 107)
##   Draws: 10 chains, each with iter = 40000; warmup = 30000; thin = 5;
##          total post-warmup draws = 20000
##
## Population-Level Effects:
##               Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## x0_maleT234       38.43      2.81    32.84    44.12 1.00    19238    17898
## x0_maleT235       45.30      0.43    44.38    45.87 1.00    18882    17888
## x0_maleT236       43.24      3.53    34.85    45.85 1.00     2880     6723
## x0_maleT237       38.94      3.38    32.75    45.44 1.00    19102    18838
## x0_maleT243       43.65      1.12    41.24    45.71 1.00    18530    16223
## x0_maleT244       43.20      2.35    37.41    45.80 1.00    19481    19070
## x0_maleT246       44.18      2.76    35.04    45.88 1.00    19543    19536
## x0_maleT247       43.02      1.47    39.85    45.11 1.00    18948    17746
## x0_maleT257       43.60      1.42    40.77    45.78 1.00    19713    18048
## x0_maleT258       37.04      2.91    32.30    42.70 1.00    19333    19224
## x0_maleT260       42.08      2.77    34.60    45.74 1.00    20401    19279
```

```
## y0_maleT234       64.32     14.94    39.46    97.75 1.00    19503    18786
## y0_maleT235      138.24     14.52   110.53   167.01 1.00    19489    19381
## y0_maleT236      167.09     25.03   130.24   232.38 1.00     3359     6019
## y0_maleT237      164.21     33.07   112.52   240.92 1.00    19943    19305
## y0_maleT243      121.32     15.37    92.67   152.70 1.00    19314    19292
## y0_maleT244       85.35     12.99    63.02   114.01 1.00    19784    18828
## y0_maleT246       21.91      8.12    10.70    41.05 1.00    19231    17884
## y0_maleT247      118.33     14.13    92.09   147.65 1.00    19899    19478
## y0_maleT257      240.49     31.50   181.59   304.96 1.00    19829    19990
## y0_maleT258       53.56     17.17    24.61    91.38 1.00    19612    17896
## y0_maleT260       93.70     19.68    58.71   135.62 1.00    19741    19501
## disp_Intercept    19.66      0.32    18.79    19.99 1.00    19700    18954
##
## 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).
## # A tibble: 10 x 5
##    chains  mean    sd     n     se
##     <int> <dbl> <dbl> <int>  <dbl>
##  1      1 -666.  4.11  2000 0.0919
##  2      2 -666.  4.16  2000 0.0931
##  3      3 -666.  4.02  2000 0.0900
##  4      4 -666.  4.01  2000 0.0897
##  5      5 -666.  4.22  2000 0.0943
##  6      6 -666.  4.03  2000 0.0900
##  7      7 -666.  4.25  2000 0.0951
##  8      8 -666.  4.16  2000 0.0931
##  9      9 -666.  4.09  2000 0.0915
## 10     10 -666.  4.10  2000 0.0916
## [1] "Plotting Trace"
## [1] "Plotting violin"
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## [1] "Plotting hist"
```

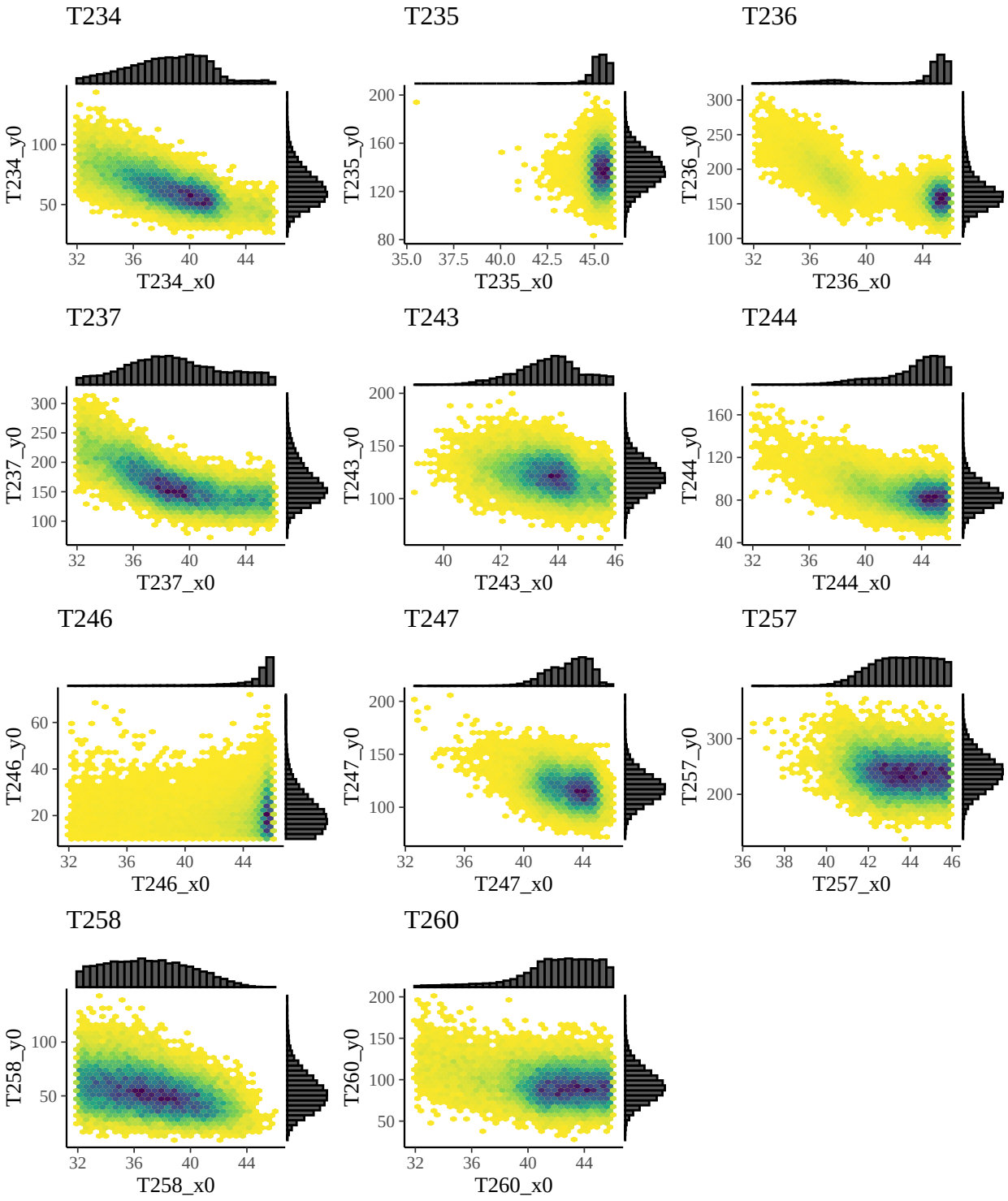



```
## [1] "Plotting Hex"
```

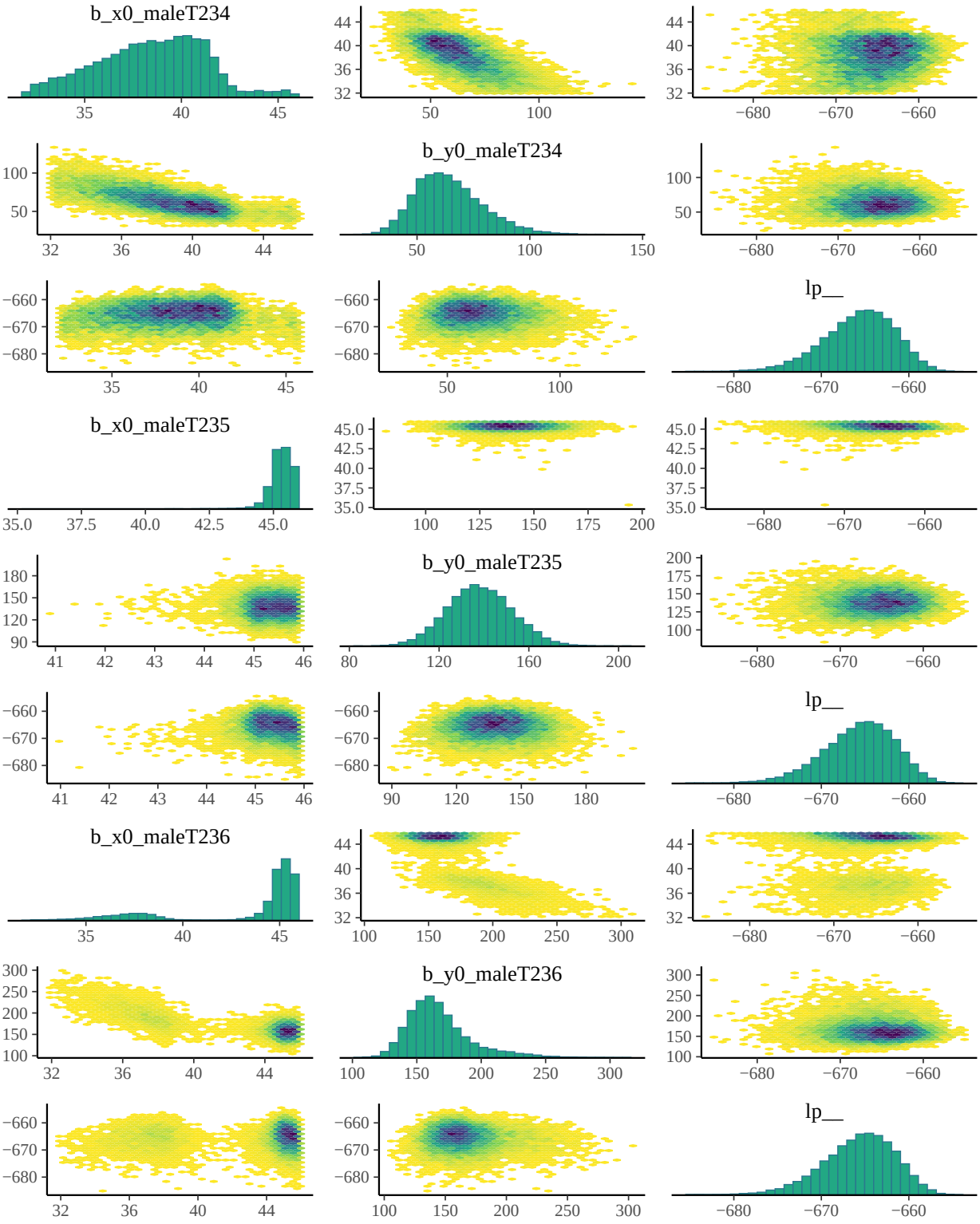
x0: individual; y0: individual; disp_flag: uniform_1; disp prior: 0.01; filter: FALSE



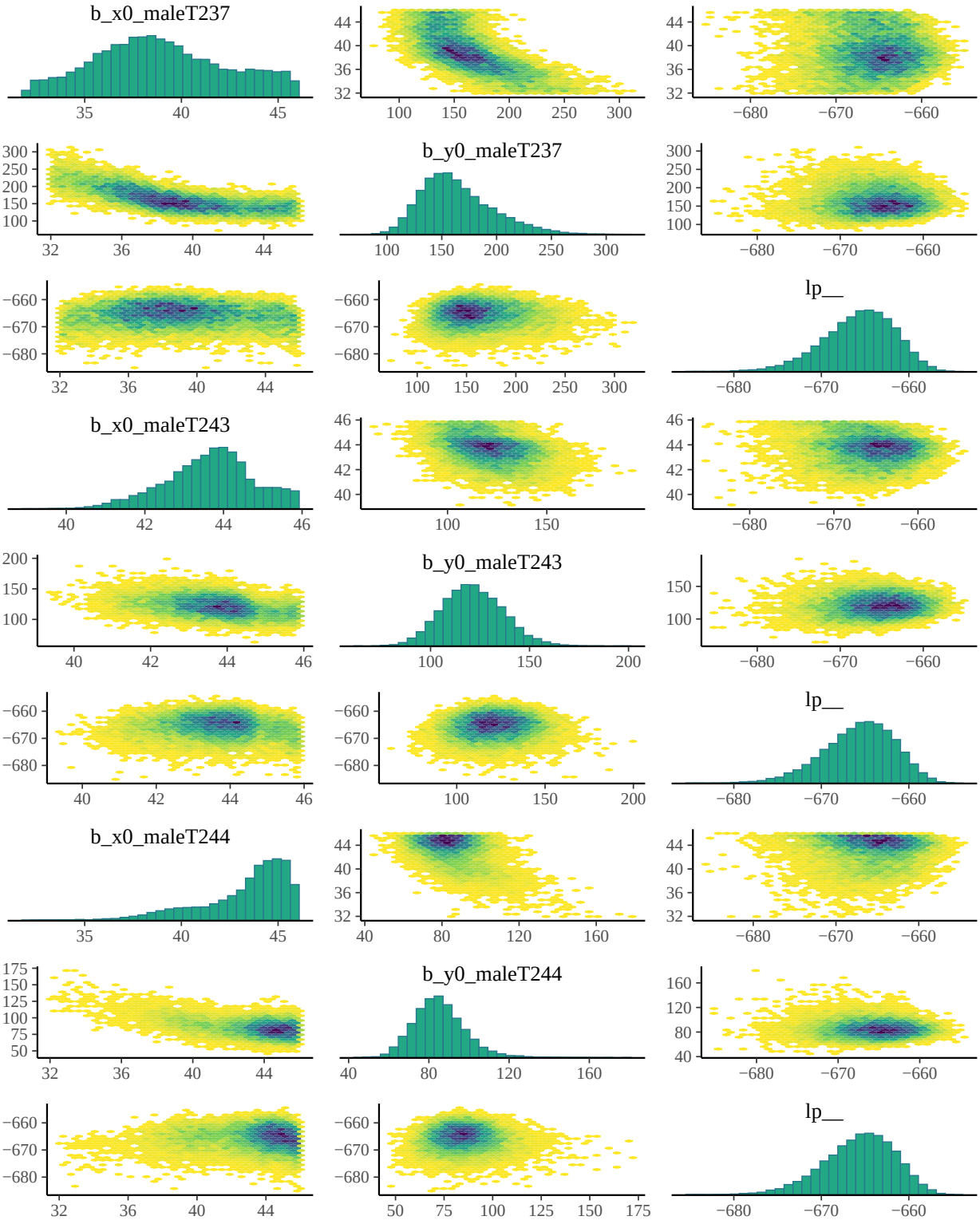
x0: individual; y0: individual; disp_flag: uniform_1; disp prior: 0.01; filter: FALSE

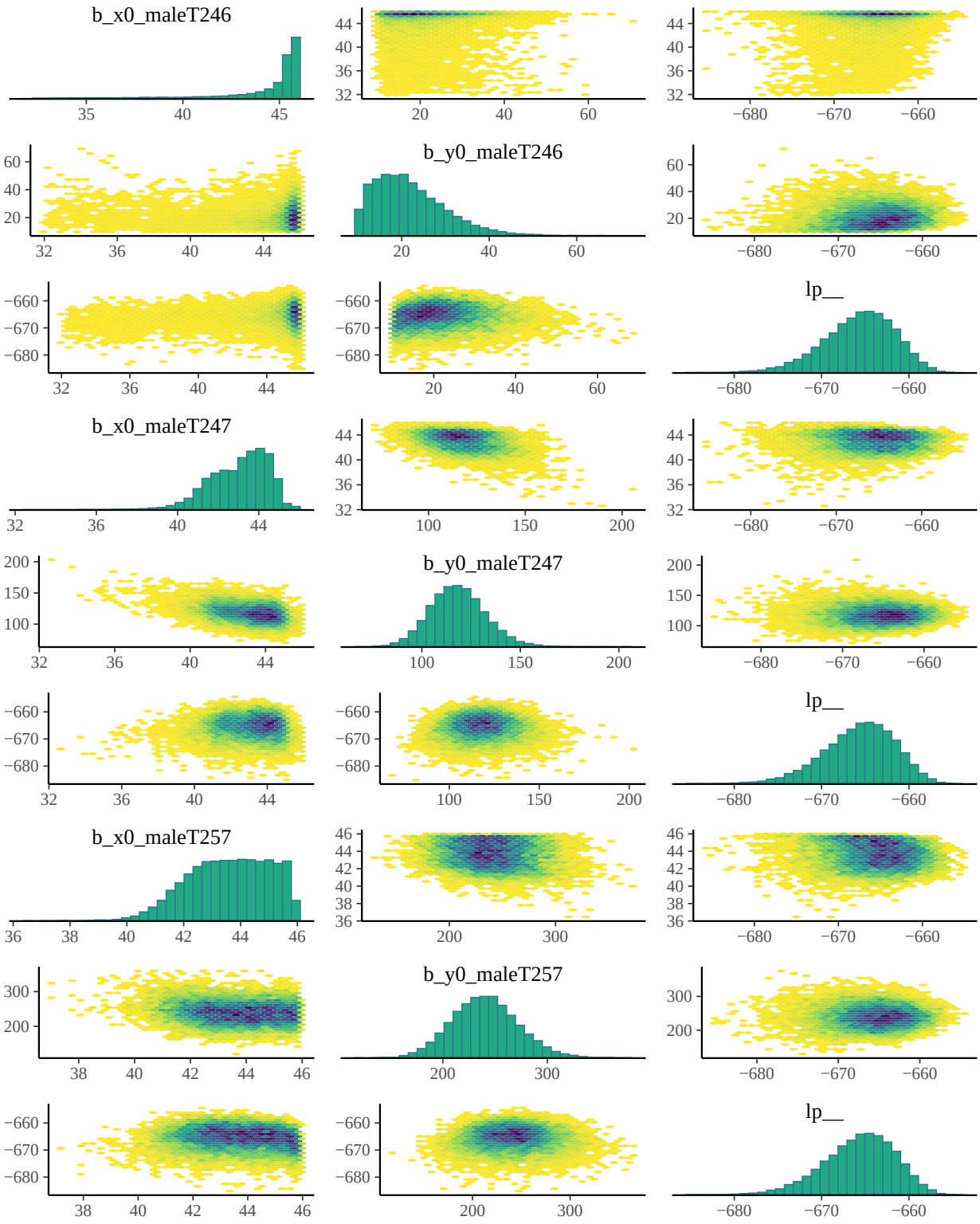



```
## [1] "Plotting Pairs"
```


x0: individual; y0: individual; disp_flag: uniform_1; disp prior: 0.01; filter: FALSE
b_x0_maleT234
b_y0_maleT234
lp__
b_x0_maleT235
b_y0_maleT235
lp__
b_x0_maleT236
b_y0_maleT236
lp__

page 1 of 4


x0: individual; y0: individual; disp_flag: uniform_1; disp prior: 0.01; filter: FALSE
b_x0_maleT237
b_y0_maleT237
lp__
b_x0_maleT243
b_y0_maleT243
lp__
b_x0_maleT244
b_y0_maleT244
lp__

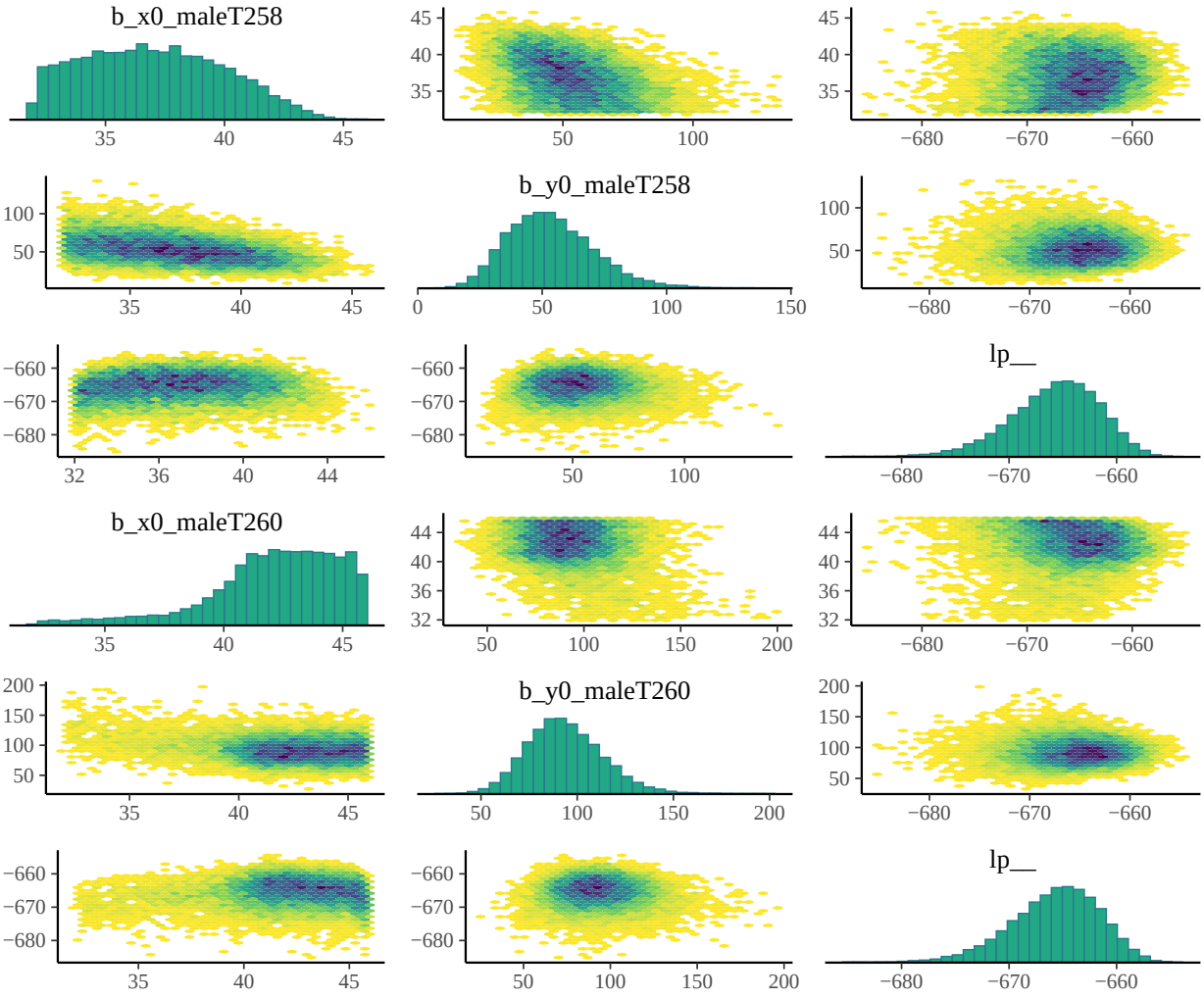


page 3 of 4

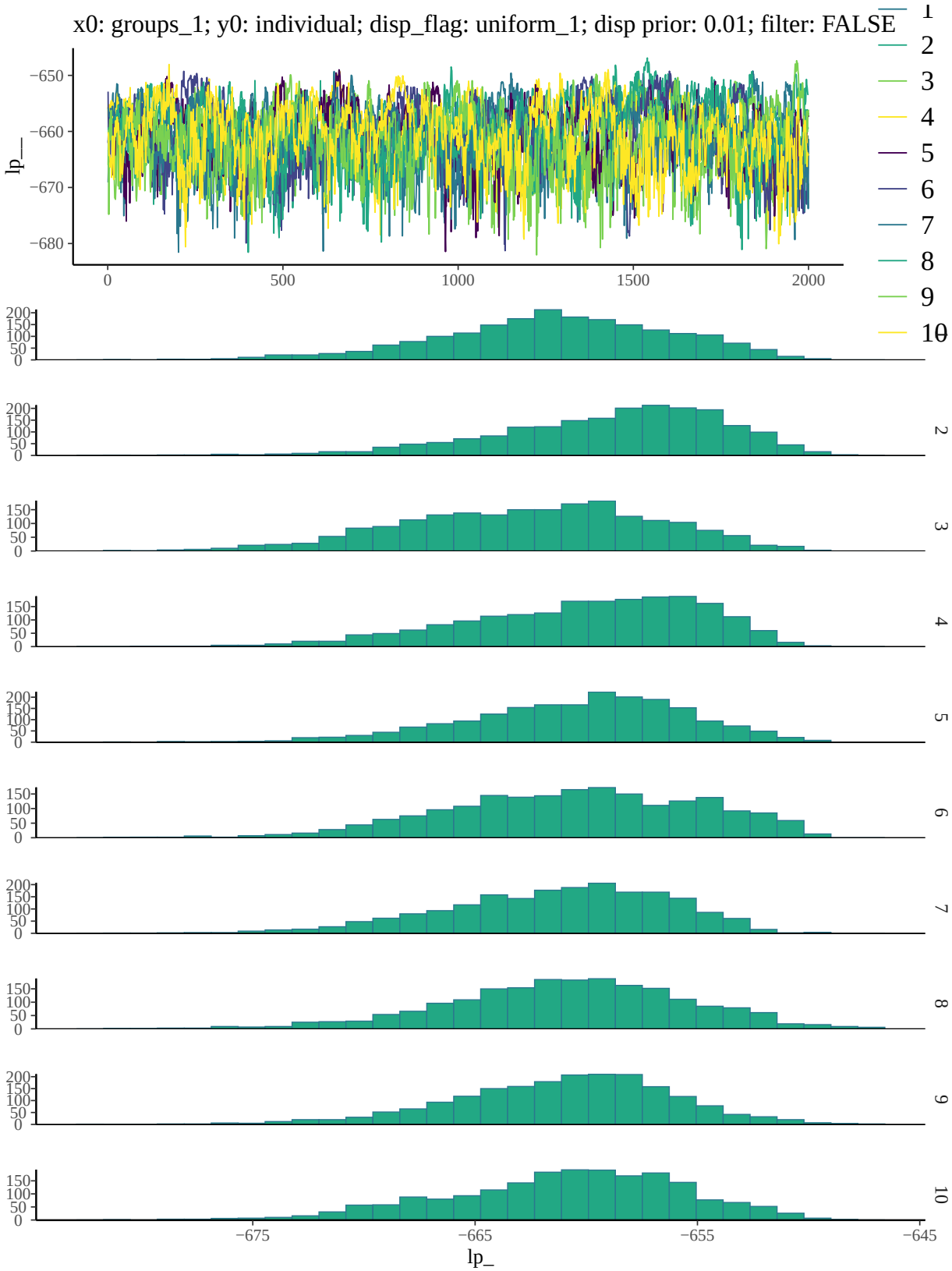
```
## \clearpage[1] "Working with Pre-existing Fits"
## [1] "`class(fit)` = brmsfit"
```

```
##
##
##
## [1] "nbinom_type1; two_piece; x0: groups_1; y0: individual; disp_flag: uniform_1; disp prior: 0.01; i
## [1] "Prior Information"
##                    prior class      coef group resp dpar nlpar lb   ub
##        uniform(32, 45.9)     b                               x0 32 45.9
##        uniform(32, 45.9)     b Intercept                     x0 32 45.9
##        uniform(10, 1000)     b                               y0 10 1000
##        uniform(10, 1000)     b  maleT234                     y0 10 1000
##        uniform(10, 1000)     b  maleT235                     y0 10 1000
##        uniform(10, 1000)     b  maleT236                     y0 10 1000
##        uniform(10, 1000)     b  maleT237                     y0 10 1000
##        uniform(10, 1000)     b  maleT243                     y0 10 1000
##        uniform(10, 1000)     b  maleT244                     y0 10 1000
##        uniform(10, 1000)     b  maleT246                     y0 10 1000
##        uniform(10, 1000)     b  maleT247                     y0 10 1000
##        uniform(10, 1000)     b  maleT257                     y0 10 1000
##        uniform(10, 1000)     b  maleT258                     y0 10 1000
##        uniform(10, 1000)     b  maleT260                     y0 10 1000
##  exponential(disp_value)     b                         disp        0   20
##  exponential(disp_value)     b Intercept               disp        0   20
##    student_t(3, 0, 66.7)    sd                               x0  0   10
##    student_t(3, 0, 66.7)    sd            male               x0  0   10
##    student_t(3, 0, 66.7)    sd Intercept  male               x0  0   10
##        source
##          user
##  (vectorized)
##          user
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
##          user
##  (vectorized)
##          user
##  (vectorized)
##  (vectorized)
## [1] "nbinom_type1; two_piece; x0: groups_1; y0: individual; disp_flag: uniform_1; disp prior: 0.01; i
## [1] "Fit Information"

## Warning: Parts of the model have not converged (some Rhats are > 1.05). Be
## careful when analysing the results! We recommend running more iterations and/or
## setting stronger priors.

## Warning: There were 19868 divergent transitions after warmup. Increasing
## adapt_delta above 0.9 may help. See
## http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
```

```
##  Family: nbinom_type1
##   Links: mu = identity; disp = identity
## Formula: y ~ two_piece(x, x0, y0)
##          x0 ~ 0 + Intercept + (1 | male)
##          disp ~ 0 + Intercept
##          y0 ~ 0 + male
##    Data: data (Number of observations: 107)
##   Draws: 10 chains, each with iter = 40000; warmup = 30000; thin = 5;
##          total post-warmup draws = 20000
##
## Group-Level Effects:
## ~male (Number of levels: 11)
##                  Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(x0_Intercept)     1.68      1.30     0.10     5.25 1.12       59       28
##
## Population-Level Effects:
##               Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## x0_Intercept     43.79      1.24    40.09    45.47 1.12       56       24
## y0_maleT234      51.87     11.24    34.09    78.69 1.08       92       70
## y0_maleT235     137.41     14.49   109.61   166.67 1.01      845     1362
## y0_maleT236     159.27     15.88   130.42   193.60 1.01      687      597
## y0_maleT237     140.76     19.32   104.21   178.85 1.01      501      936
## y0_maleT243     120.17     15.44    90.61   151.46 1.01      618      794
## y0_maleT244      83.87     10.89    63.33   106.05 1.02      797     1573
## y0_maleT246      22.38      8.00    11.07    40.98 1.01     1107     1263
## y0_maleT247     116.44     13.67    91.37   144.63 1.01      721      970
## y0_maleT257     237.97     31.17   179.44   300.62 1.01      617      746
## y0_maleT258      39.40     13.55    17.03    69.78 1.02      556     1324
## y0_maleT260      90.13     18.20    55.92   127.40 1.01      908     1242
## disp_Intercept   19.67      0.32    18.82    19.99 1.01      727      784
##
## 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).
## # A tibble: 10 x 5
##    chains  mean    sd     n    se
##     <int> <dbl> <dbl> <int> <dbl>
##  1      1 -661.  5.20  2000 0.116
##  2      2 -659.  5.06  2000 0.113
##  3      3 -663.  5.58  2000 0.125
##  4      4 -660.  5.23  2000 0.117
##  5      5 -660.  4.95  2000 0.111
##  6      6 -661.  5.63  2000 0.126
##  7      7 -661.  4.95  2000 0.111
##  8      8 -661.  5.36  2000 0.120
##  9      9 -661.  4.89  2000 0.109
## 10     10 -661.  5.26  2000 0.118
## [1] "Plotting Trace"
## [1] "Plotting violin"
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

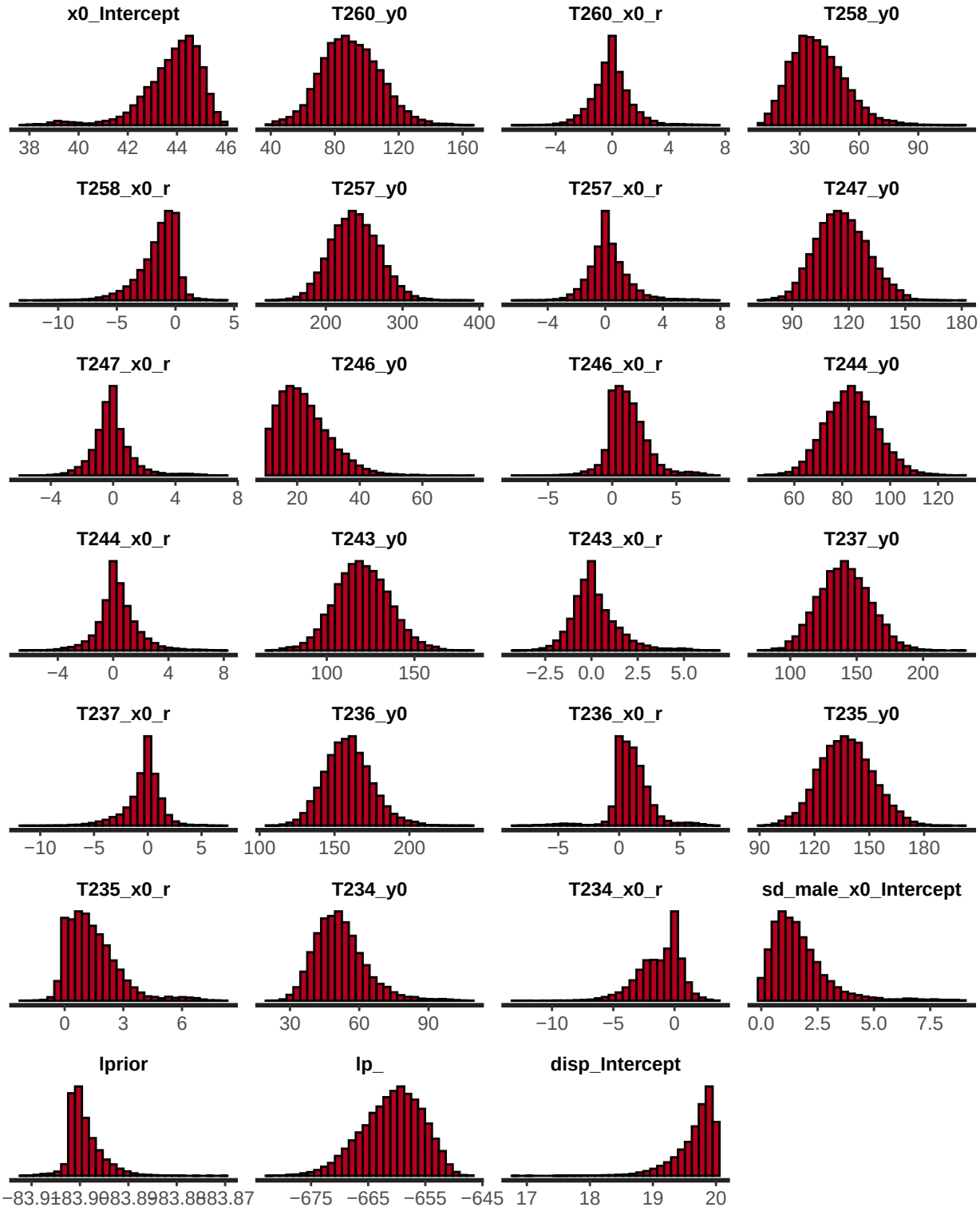



```
## [1] "Plotting hist"
```

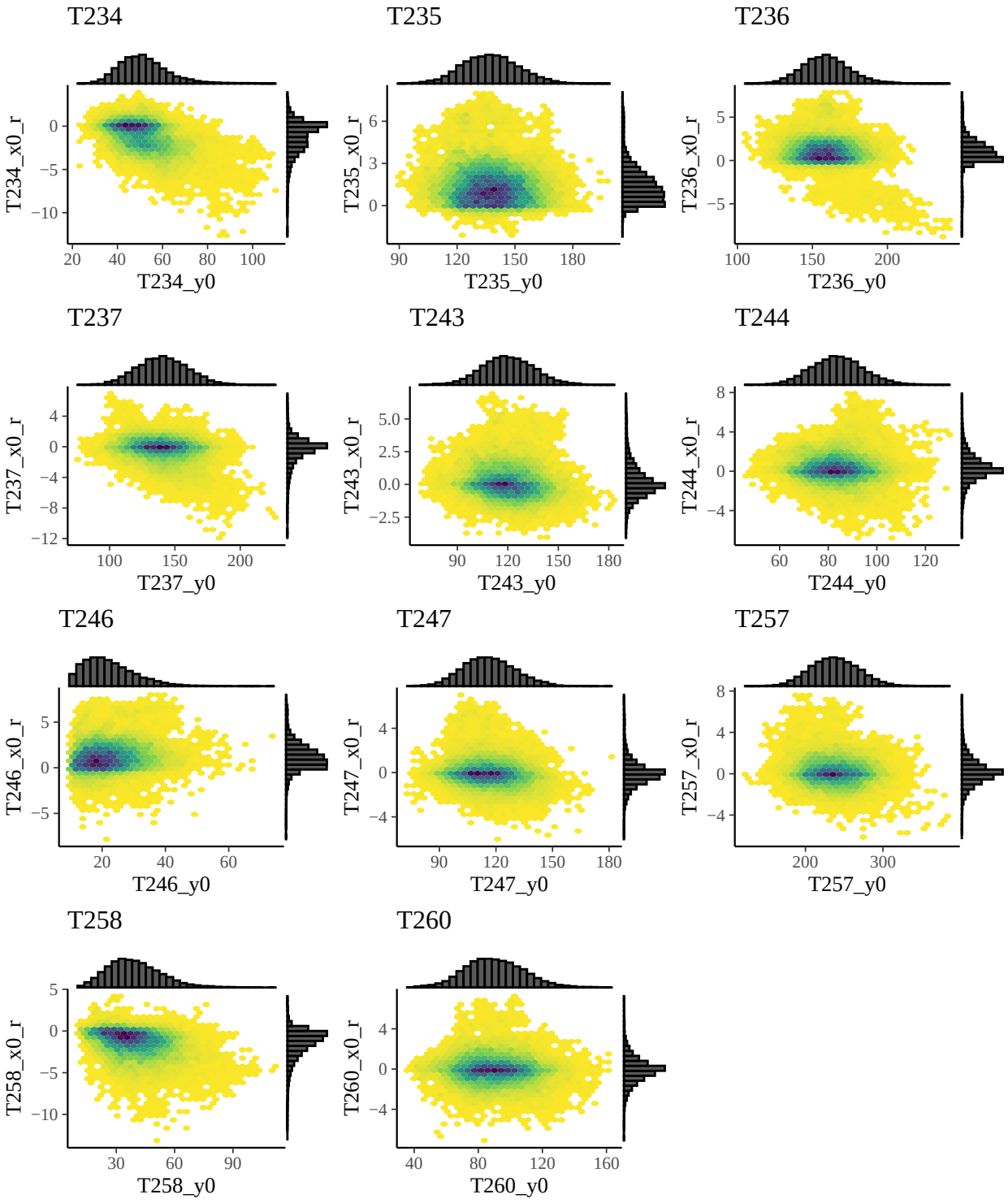



```
## [1] "Plotting Hex"
```

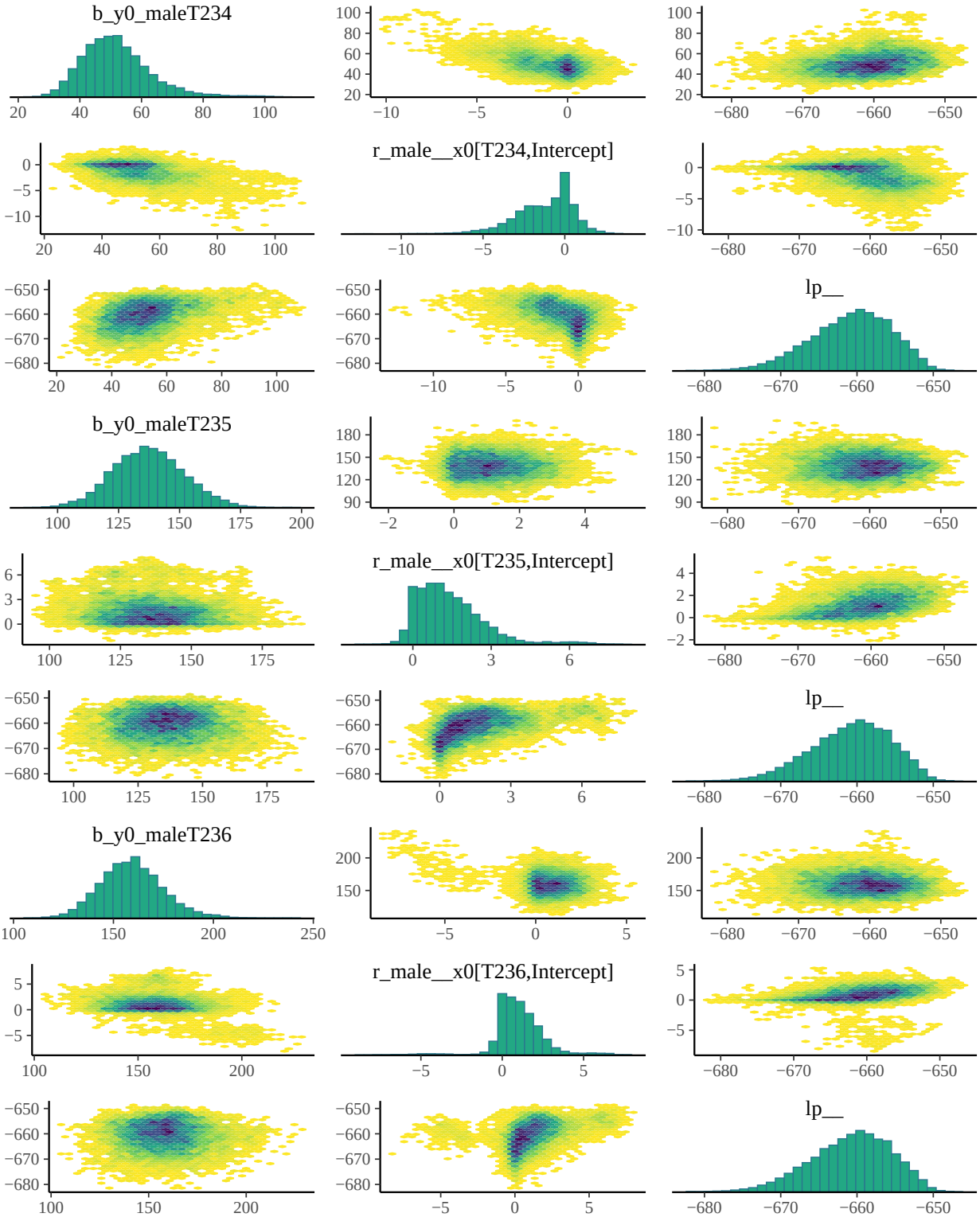
x0: groups_1; y0: individual; disp_flag: uniform_1; disp prior: 0.01; filter: FALSE

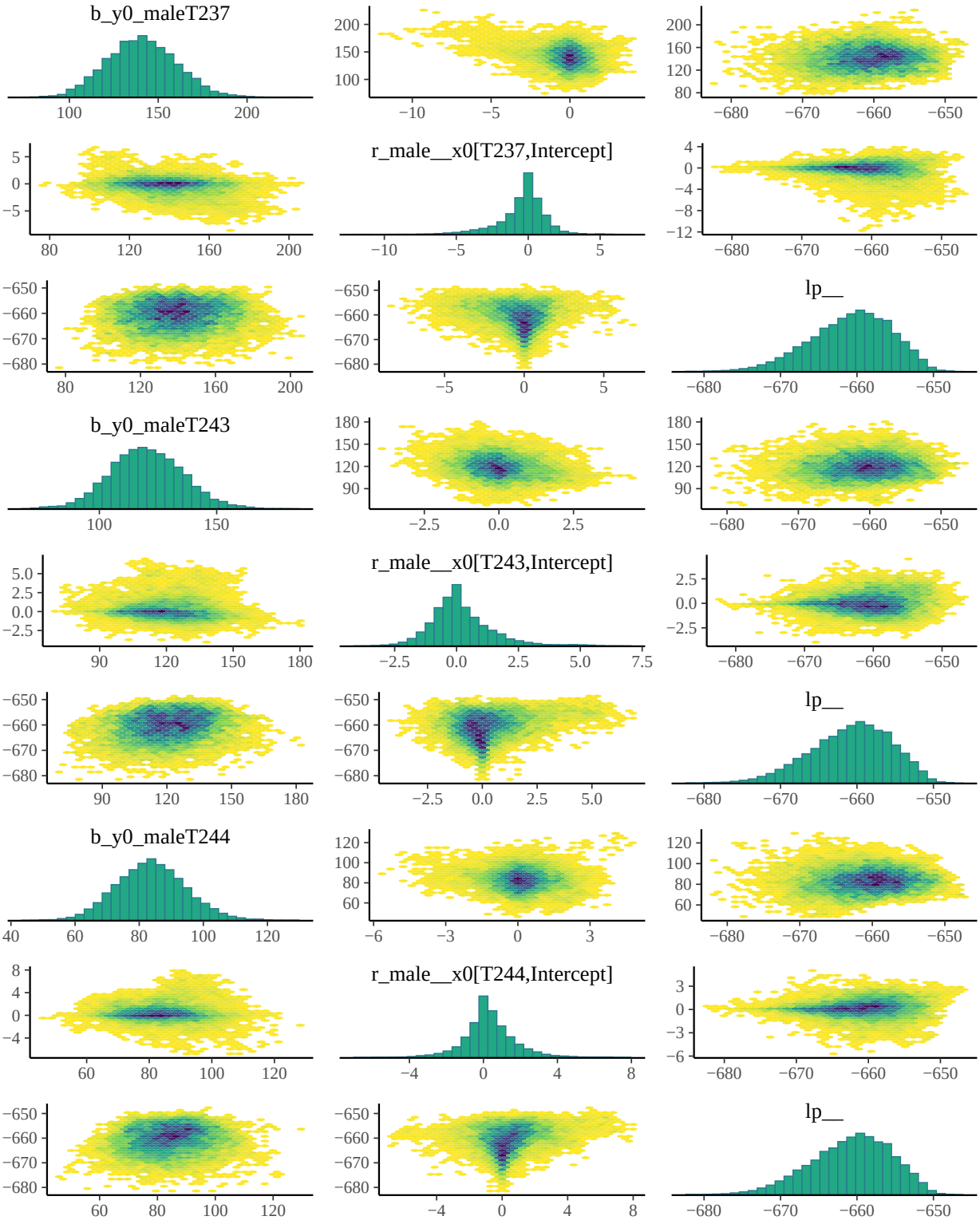


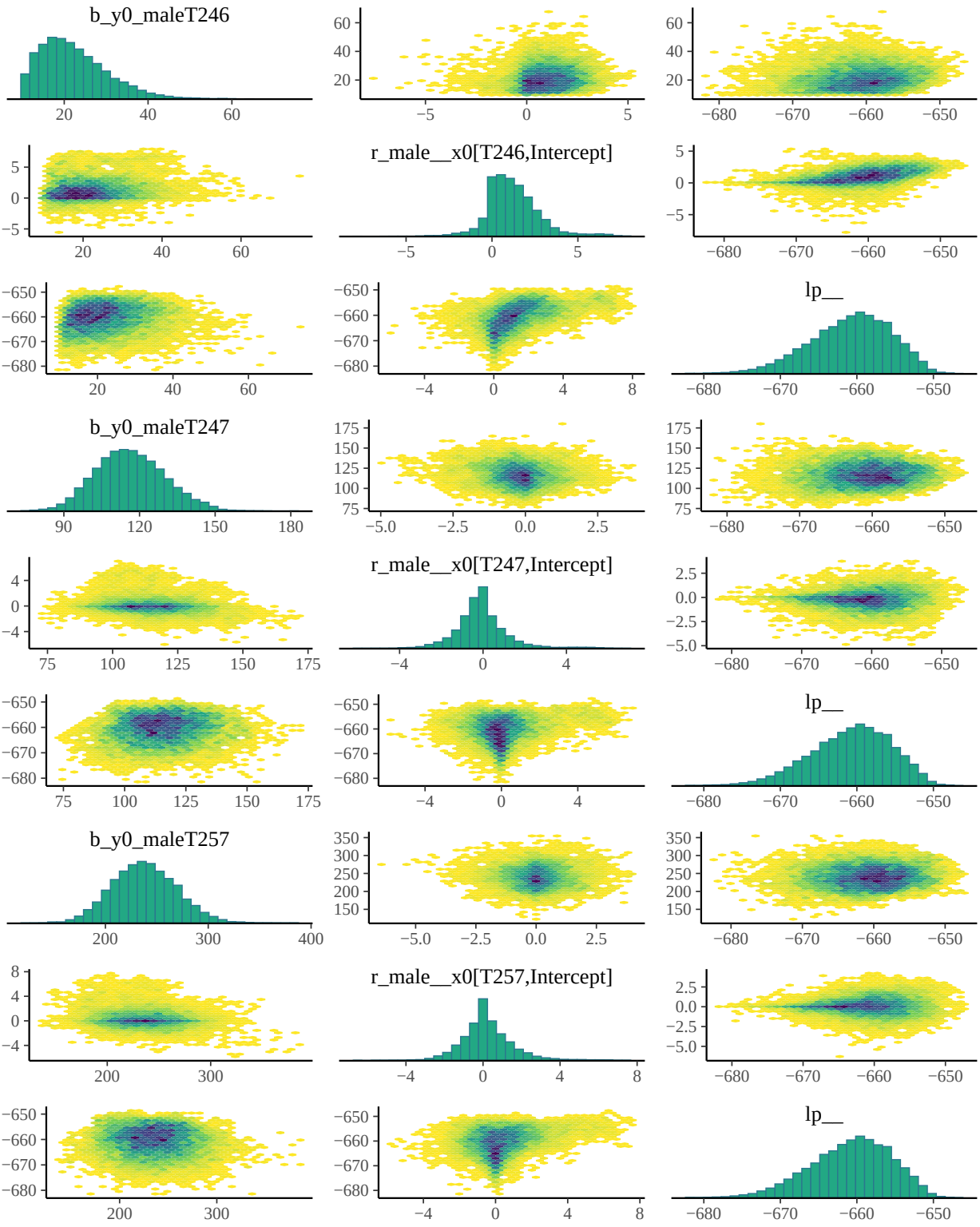
x0: groups_1; y0: individual; disp_flag: uniform_1; disp prior: 0.01; filter: FALSE




```
## [1] "Plotting Pairs"
```


x0: groups_1; y0: individual; disp_flag: uniform_1; disp prior: 0.01; filter: FALSE
b_y0_maleT234
r_male__x0[T234,Intercept]
lp__
b_y0_maleT235
r_male__x0[T235,Intercept]
lp__
b_y0_maleT236
r_male__x0[T236,Intercept]
lp__

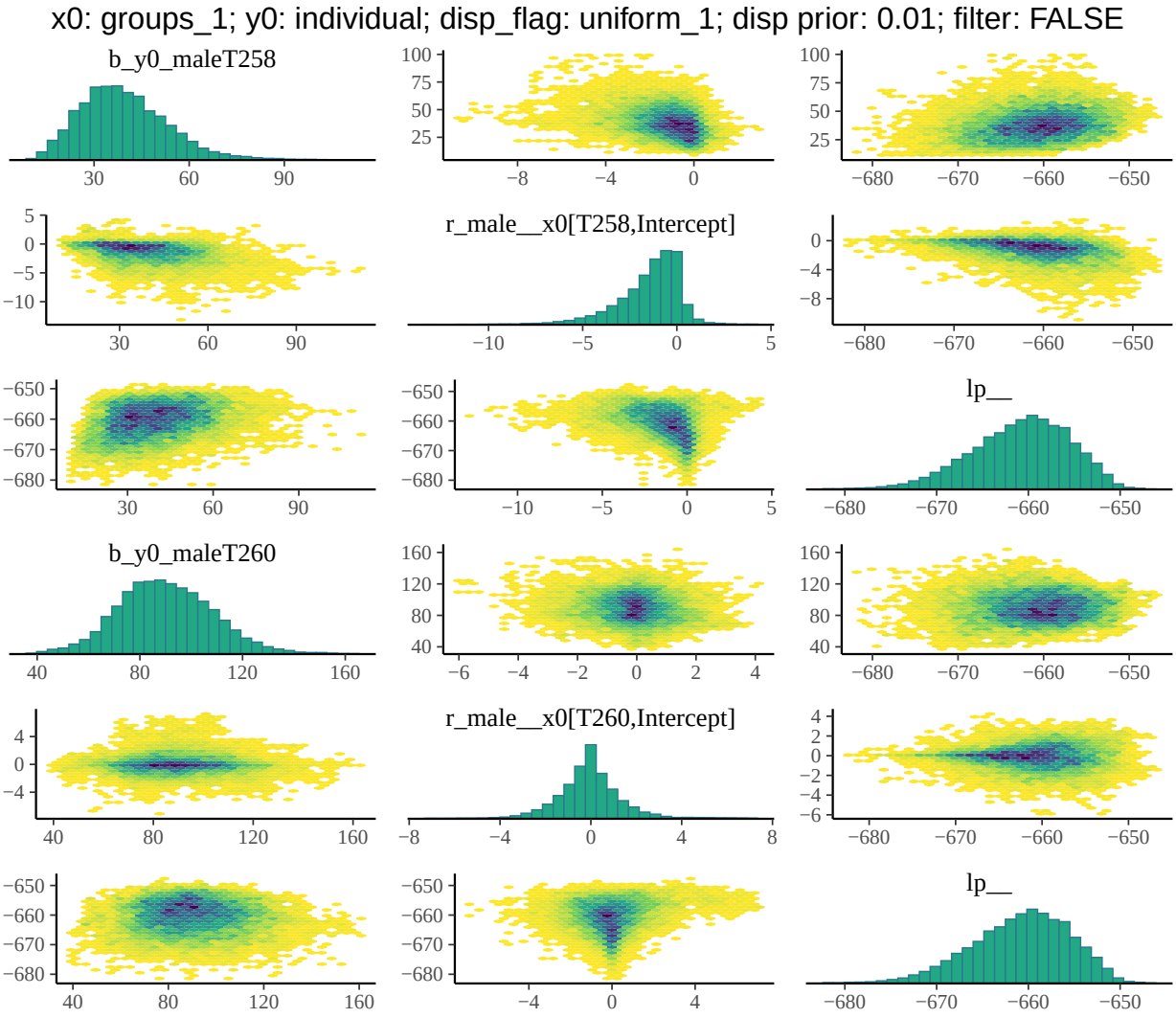

x0: groups_1; y0: individual; disp_flag: uniform_1; disp prior: 0.01; filter: FALSE
b_y0_maleT237
r_male__x0[T237,Intercept]
lp__
b_y0_maleT243
r_male__x0[T243,Intercept]
lp__
b_y0_maleT244
r_male__x0[T244,Intercept]
lp__



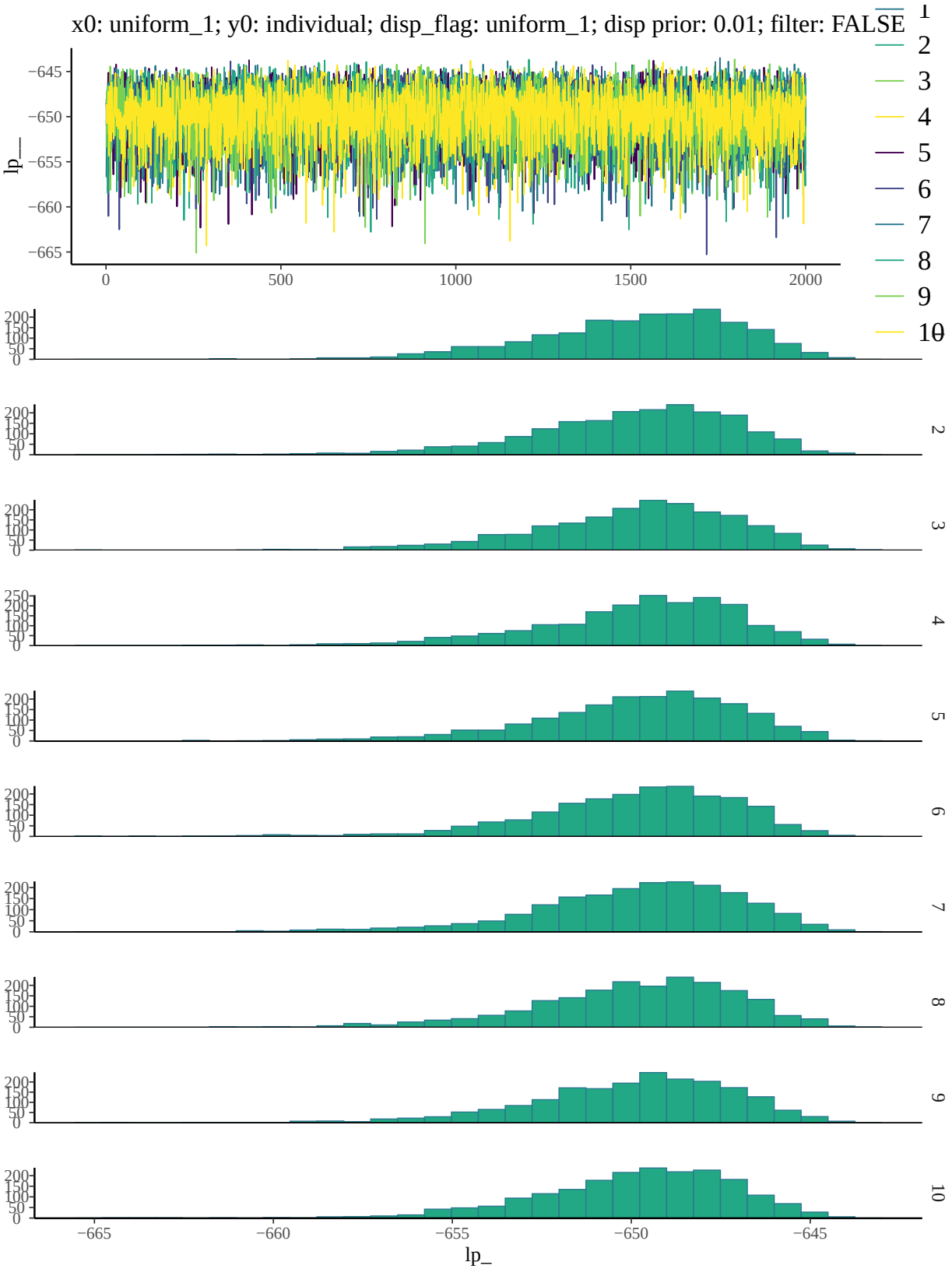

```
## \clearpage[1] "Working with Pre-existing Fits"
## [1] "`class(fit)` = brmsfit"
```

```
##
##
##
## [1] "nbinom_type1; two_piece; x0: uniform_1; y0: individual; disp_flag: uniform_1; disp prior: 0.01;
## [1] "Prior Information"
##                    prior class      coef group resp dpar nlpar lb   ub
##        uniform(32, 45.9)     b                               x0 32 45.9
##        uniform(32, 45.9)     b Intercept                     x0 32 45.9
##        uniform(10, 1000)     b                               y0 10 1000
##        uniform(10, 1000)     b  maleT234                     y0 10 1000
##        uniform(10, 1000)     b  maleT235                     y0 10 1000
##        uniform(10, 1000)     b  maleT236                     y0 10 1000
##        uniform(10, 1000)     b  maleT237                     y0 10 1000
##        uniform(10, 1000)     b  maleT243                     y0 10 1000
##        uniform(10, 1000)     b  maleT244                     y0 10 1000
##        uniform(10, 1000)     b  maleT246                     y0 10 1000
##        uniform(10, 1000)     b  maleT247                     y0 10 1000
##        uniform(10, 1000)     b  maleT257                     y0 10 1000
##        uniform(10, 1000)     b  maleT258                     y0 10 1000
##        uniform(10, 1000)     b  maleT260                     y0 10 1000
##  exponential(disp_value)     b                         disp        0   20
##  exponential(disp_value)     b Intercept               disp        0   20
##        source
##          user
##  (vectorized)
##          user
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
##          user
##  (vectorized)
## [1] "nbinom_type1; two_piece; x0: uniform_1; y0: individual; disp_flag: uniform_1; disp prior: 0.01;
## [1] "Fit Information"
##  Family: nbinom_type1
##   Links: mu = identity; disp = identity
## Formula: y ~ two_piece(x, x0, y0)
##          x0 ~ 0 + Intercept
##          disp ~ 0 + Intercept
##          y0 ~ 0 + male
##    Data: data (Number of observations: 107)
##   Draws: 10 chains, each with iter = 40000; warmup = 30000; thin = 5;
##          total post-warmup draws = 20000
##
## Population-Level Effects:
##               Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## x0_Intercept      45.24      0.49    44.15    45.88 1.00    17597    19069
```

```
## y0_maleT234       46.21      7.88    31.55    62.47 1.00    19419    19168
## y0_maleT235      138.07     14.46   110.66   167.49 1.00    19339    19779
## y0_maleT236      158.12     15.55   128.22   189.07 1.00    19986    20041
## y0_maleT237      138.98     18.23   104.31   176.19 1.00    19656    19496
## y0_maleT243      112.66     13.43    87.56   140.31 1.00    20381    19701
## y0_maleT244       81.89     10.71    61.78   103.61 1.00    19976    20124
## y0_maleT246       21.85      8.01    10.73    40.63 1.00    18411    18164
## y0_maleT247      109.47     12.56    85.71   135.15 1.00    19508    19582
## y0_maleT257      237.84     30.68   179.48   299.95 1.00    18997    19782
## y0_maleT258       29.70      9.82    13.32    51.42 1.00    18511    16916
## y0_maleT260       91.27     18.32    57.96   129.17 1.00    20171    19360
## disp_Intercept    19.69      0.30    18.90    19.99 1.00    19431    18395
##
## 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).
## # A tibble: 10 x 5
##    chains  mean    sd     n     se
##     <int> <dbl> <dbl> <int>  <dbl>
##  1      1 -650.  2.77  2000 0.0620
##  2      2 -650.  2.81  2000 0.0628
##  3      3 -650.  2.79  2000 0.0623
##  4      4 -650.  2.78  2000 0.0621
##  5      5 -650.  2.83  2000 0.0633
##  6      6 -650.  2.82  2000 0.0631
##  7      7 -650.  2.85  2000 0.0637
##  8      8 -650.  2.78  2000 0.0621
##  9      9 -650.  2.78  2000 0.0622
## 10     10 -650.  2.70  2000 0.0605
## [1] "Plotting Trace"
## [1] "Plotting violin"
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

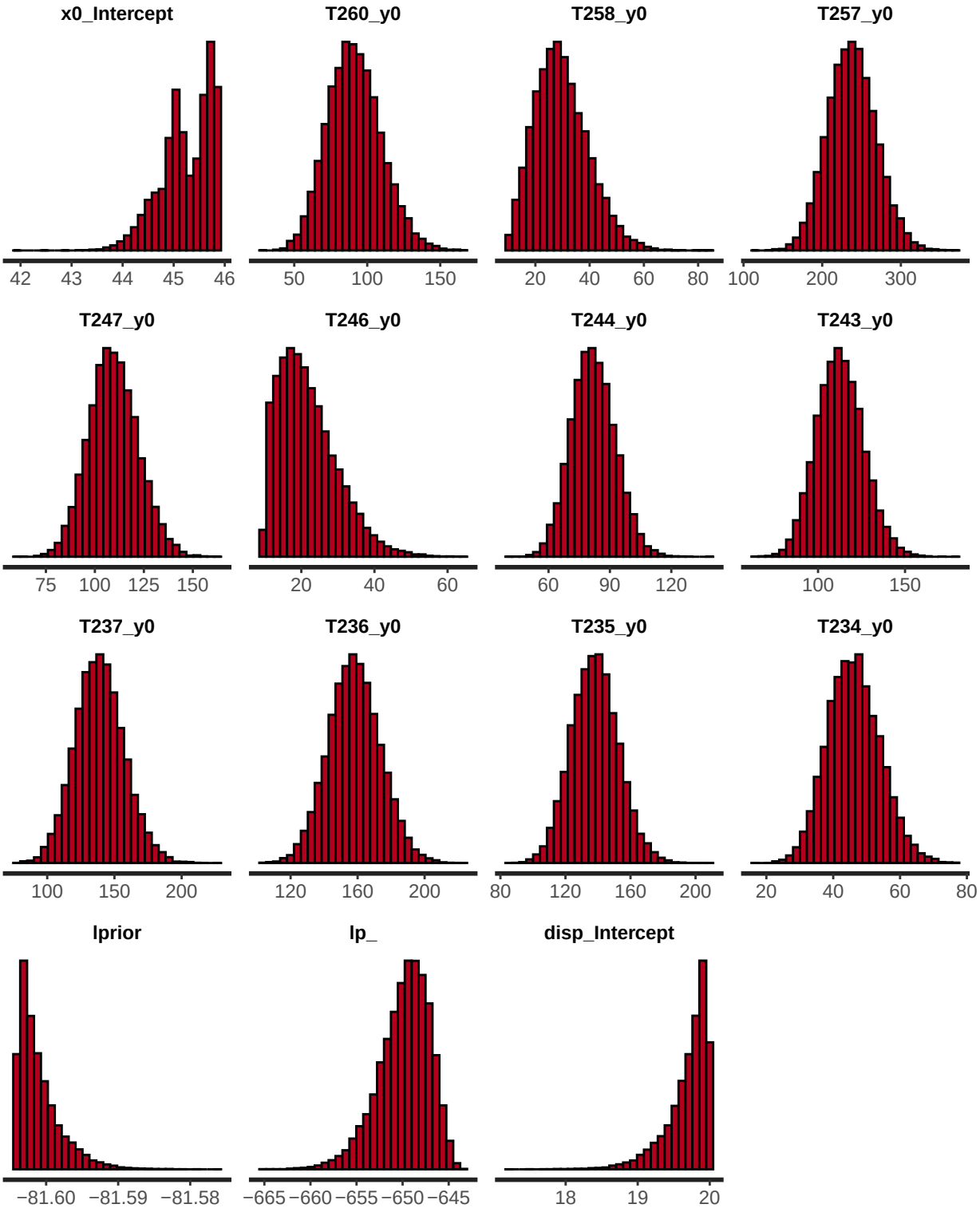



```
## [1] "Plotting hist"
```

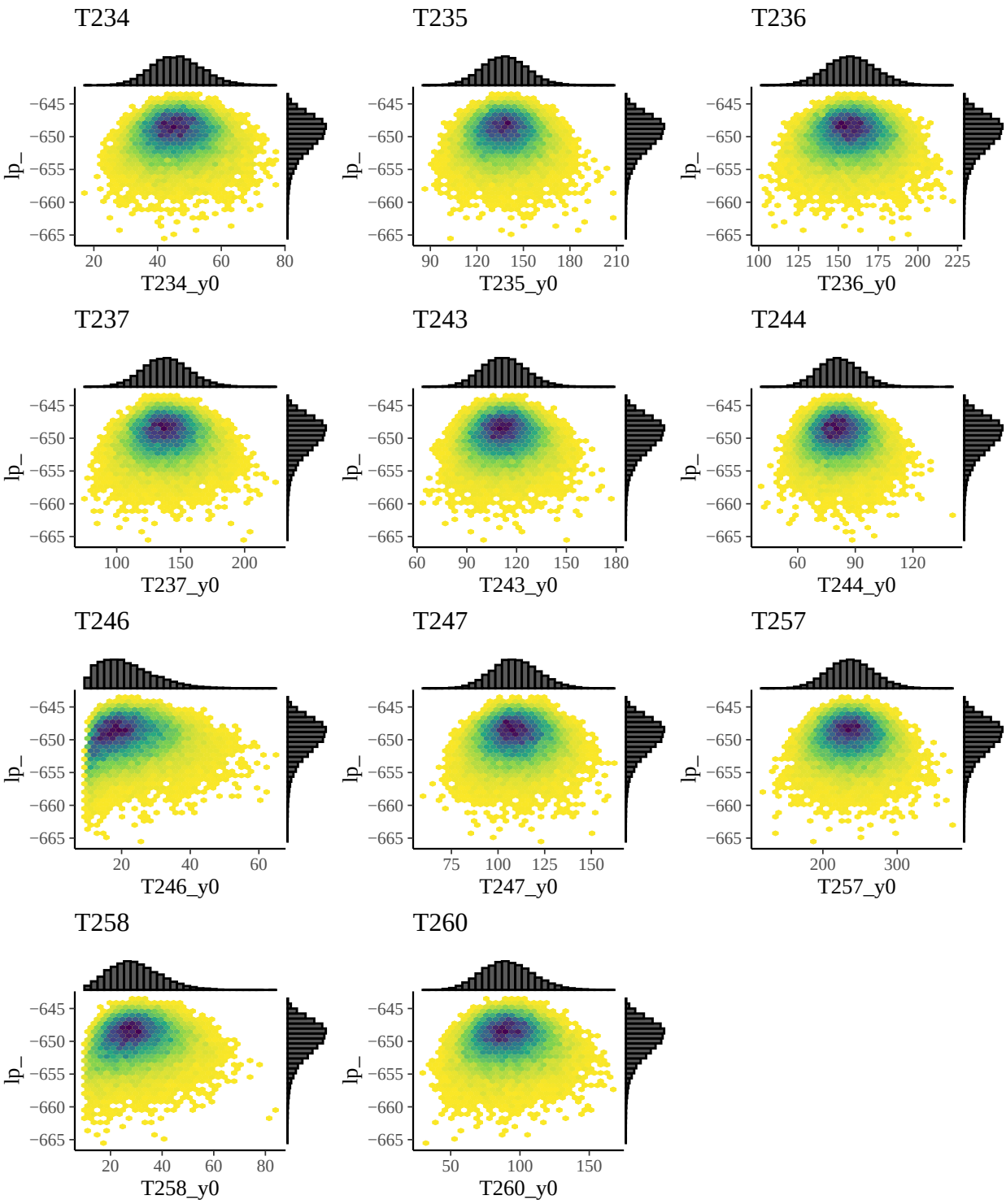



```
## [1] "Plotting Hex"
```

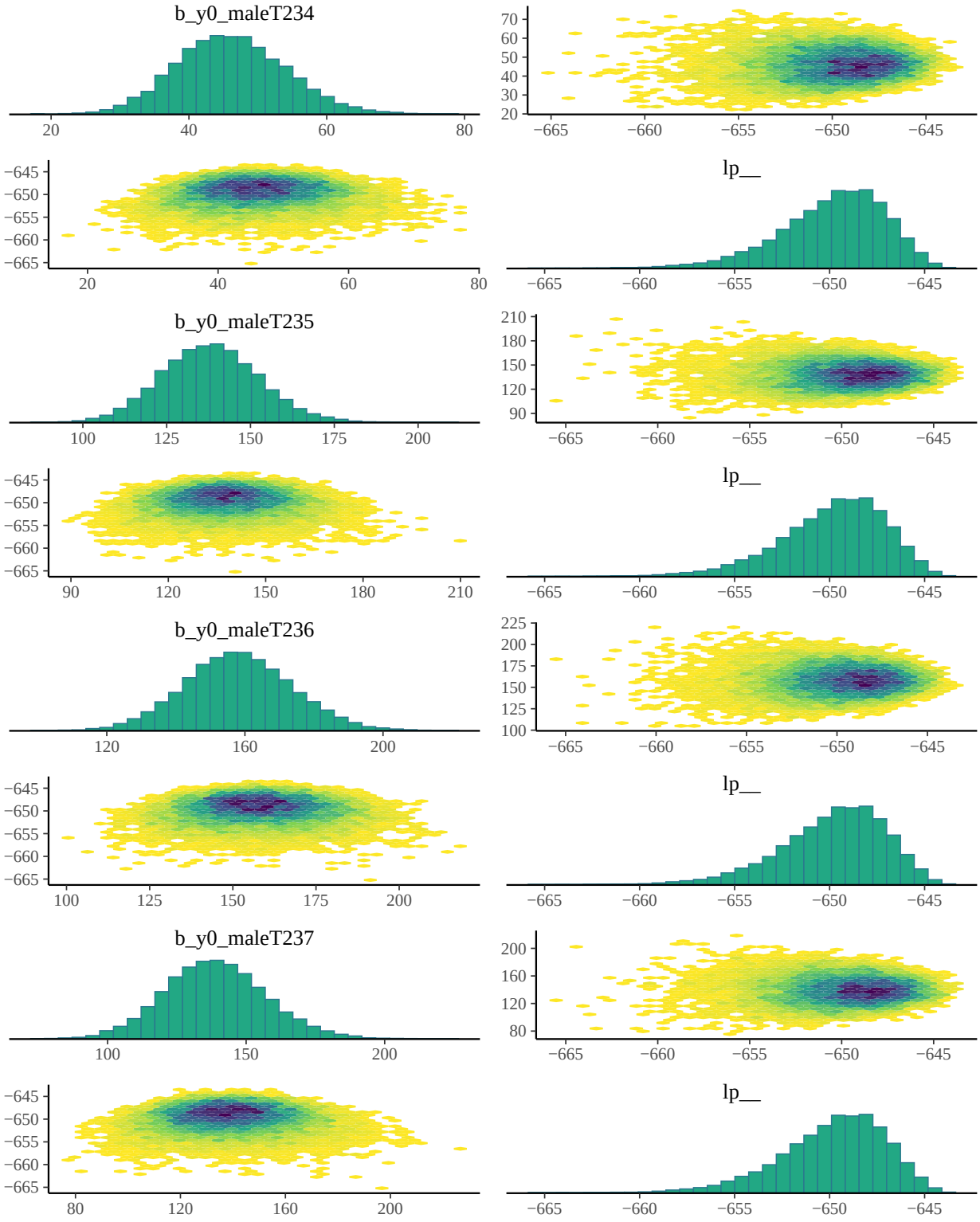
x0: uniform_1; y0: individual; disp_flag: uniform_1; disp prior: 0.01; filter: FALSE

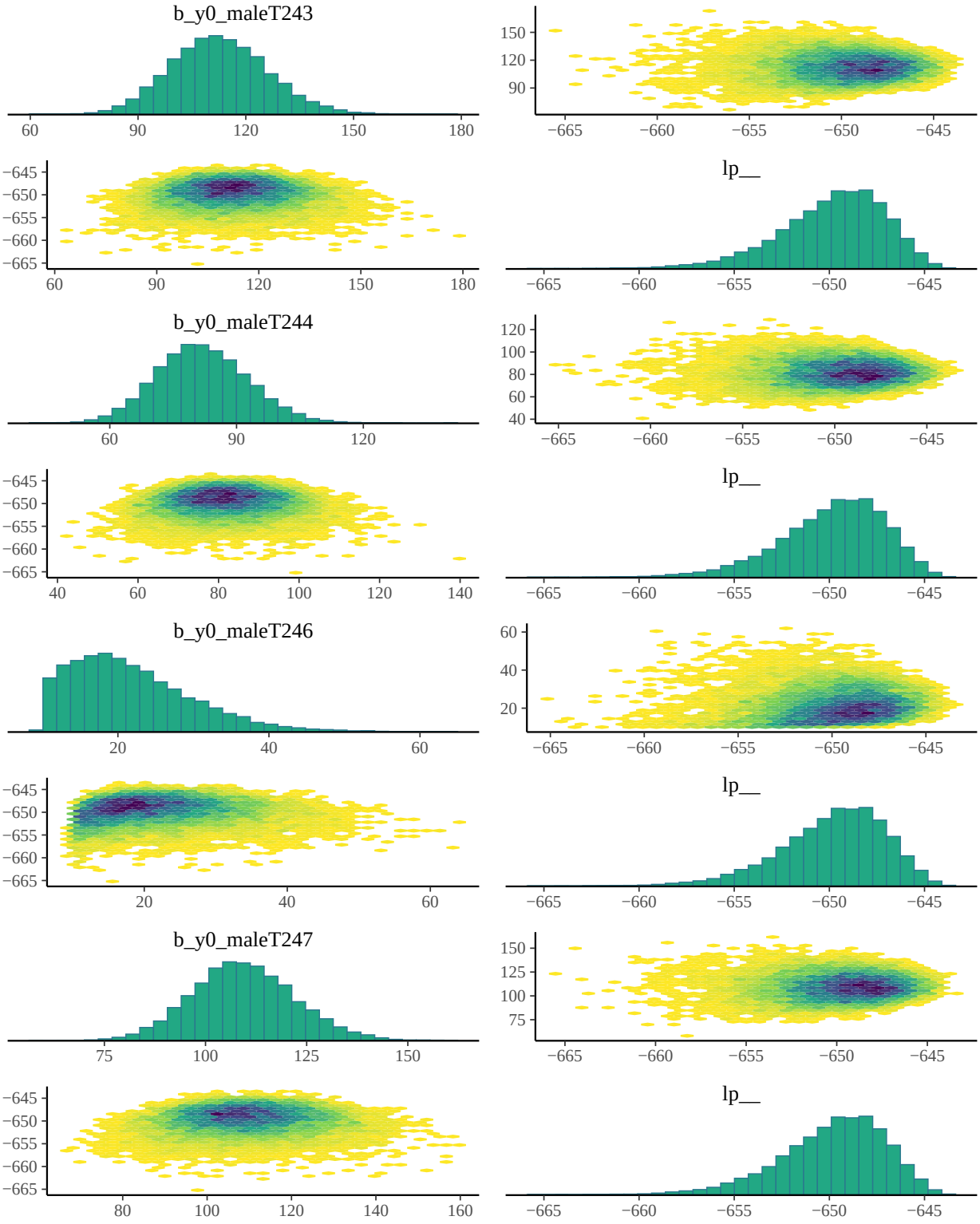


x0: uniform_1; y0: individual; disp_flag: uniform_1; disp prior: 0.01; filter: FALSE




```
## [1] "Plotting Pairs"
```

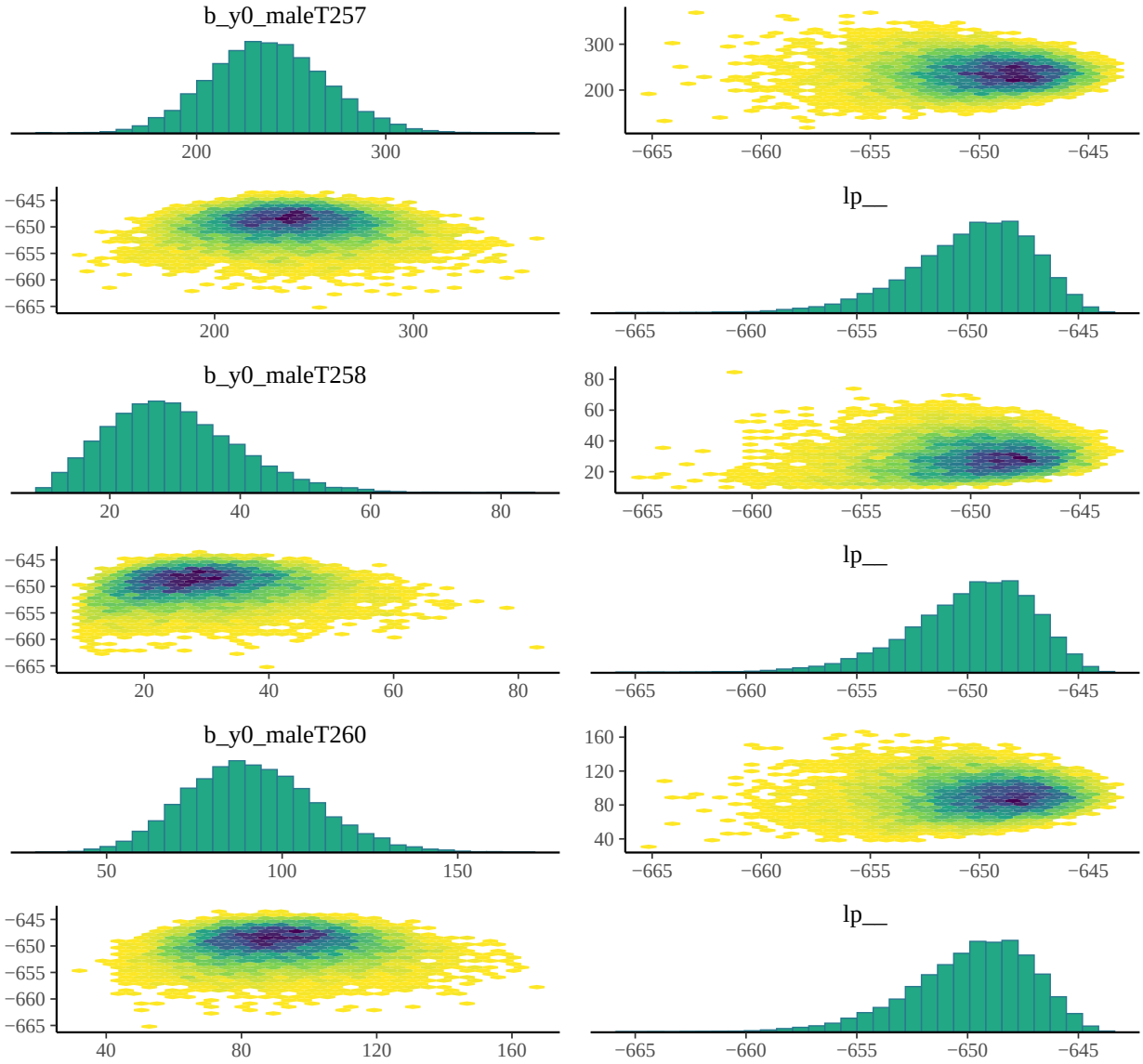

x0: uniform_1; y0: individual; disp_flag: uniform_1; disp prior: 0.01; filter: FALSE
b_y0_maleT234
lp__
b_y0_maleT235
lp__
b_y0_maleT236
lp__
b_y0_maleT237
lp__



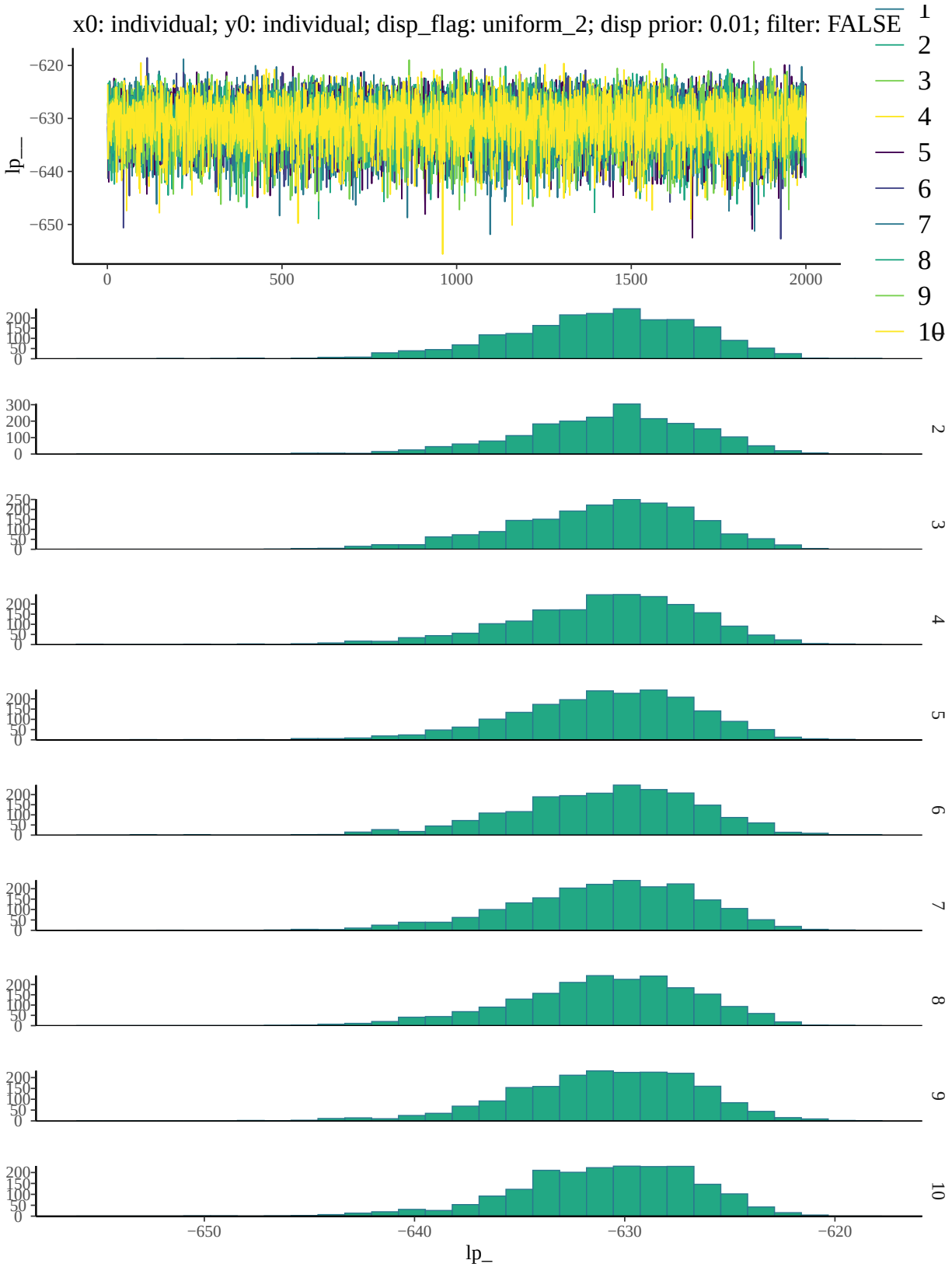

```
## \clearpage[1] "Working with Pre-existing Fits"
## [1] "`class(fit)` = brmsfit"
```

```
##
##
##
## [1] "nbinom_type1; two_piece; x0: individual; y0: individual; disp_flag: uniform_2; disp prior: 0.01
## [1] "Prior Information"
##              prior class        coef group resp dpar nlpar lb   ub       source
##  uniform(32, 45.9)     b                                  x0 32 45.9         user
##  uniform(32, 45.9)     b    maleT234                      x0 32 45.9 (vectorized)
##  uniform(32, 45.9)     b    maleT235                      x0 32 45.9 (vectorized)
##  uniform(32, 45.9)     b    maleT236                      x0 32 45.9 (vectorized)
##  uniform(32, 45.9)     b    maleT237                      x0 32 45.9 (vectorized)
##  uniform(32, 45.9)     b    maleT243                      x0 32 45.9 (vectorized)
##  uniform(32, 45.9)     b    maleT244                      x0 32 45.9 (vectorized)
##  uniform(32, 45.9)     b    maleT246                      x0 32 45.9 (vectorized)
##  uniform(32, 45.9)     b    maleT247                      x0 32 45.9 (vectorized)
##  uniform(32, 45.9)     b    maleT257                      x0 32 45.9 (vectorized)
##  uniform(32, 45.9)     b    maleT258                      x0 32 45.9 (vectorized)
##  uniform(32, 45.9)     b    maleT260                      x0 32 45.9 (vectorized)
##  uniform(10, 1000)     b                                  y0 10 1000         user
##  uniform(10, 1000)     b    maleT234                      y0 10 1000 (vectorized)
##  uniform(10, 1000)     b    maleT235                      y0 10 1000 (vectorized)
##  uniform(10, 1000)     b    maleT236                      y0 10 1000 (vectorized)
##  uniform(10, 1000)     b    maleT237                      y0 10 1000 (vectorized)
##  uniform(10, 1000)     b    maleT243                      y0 10 1000 (vectorized)
##  uniform(10, 1000)     b    maleT244                      y0 10 1000 (vectorized)
##  uniform(10, 1000)     b    maleT246                      y0 10 1000 (vectorized)
##  uniform(10, 1000)     b    maleT247                      y0 10 1000 (vectorized)
##  uniform(10, 1000)     b    maleT257                      y0 10 1000 (vectorized)
##  uniform(10, 1000)     b    maleT258                      y0 10 1000 (vectorized)
##  uniform(10, 1000)     b    maleT260                      y0 10 1000 (vectorized)
##             (flat)     b                          disp                   default
##             (flat)     b disp_group1              disp               (vectorized)
##             (flat)     b disp_group2              disp               (vectorized)
## [1] "nbinom_type1; two_piece; x0: individual; y0: individual; disp_flag: uniform_2; disp prior: 0.01
## [1] "Fit Information"
##  Family: nbinom_type1
##   Links: mu = identity; disp = identity
## Formula: y ~ two_piece(x, x0, y0)
##          x0 ~ 0 + male
##          disp ~ 0 + disp_group
##          y0 ~ 0 + male
##    Data: data (Number of observations: 107)
##   Draws: 10 chains, each with iter = 40000; warmup = 30000; thin = 5;
##          total post-warmup draws = 20000
##
## Population-Level Effects:
##                 Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## x0_maleT234        38.30      2.98    32.71    44.56 1.00    19154    18163
## x0_maleT235        43.24      2.98    33.93    45.83 1.00    18541    18337
## x0_maleT236        42.31      4.00    33.79    45.84 1.00     6698    14869
## x0_maleT237        38.91      3.56    32.56    45.47 1.00    19012    18628
## x0_maleT243        43.41      1.89    38.82    45.79 1.00    19795    18654
## x0_maleT244        42.77      2.68    36.01    45.79 1.00    19352    18760
## x0_maleT246        43.72      3.09    34.43    45.88 1.00    19241    18967
```

```
## x0_maleT247         42.86      1.78    38.55    45.32 1.00    19648    18267
## x0_maleT257         43.43      1.60    40.04    45.79 1.00    18870    18159
## x0_maleT258         37.09      3.01    32.28    43.02 1.00    19678    18476
## x0_maleT260         41.54      3.18    33.41    45.72 1.00    19952    19186
## y0_maleT234         69.70     17.78    40.57   109.50 1.00    19129    18989
## y0_maleT235        225.84     70.18   128.49   402.45 1.00    18154    15358
## y0_maleT236        176.55     31.58   130.64   255.92 1.00     8432    15033
## y0_maleT237        171.16     37.96   111.15   259.14 1.00    18951    19434
## y0_maleT243        186.86     56.52   103.09   324.51 1.00    18308    17466
## y0_maleT244         91.27     16.57    63.84   130.31 1.00    19752    18605
## y0_maleT246         23.99      9.61    10.90    46.89 1.00    18785    17988
## y0_maleT247        124.04     17.88    93.12   162.60 1.00    20019    19378
## y0_maleT257        246.87     38.35   176.28   328.10 1.00    20107    19650
## y0_maleT258         58.69     20.70    25.08   105.81 1.00    19603    18660
## y0_maleT260         99.97     24.52    57.80   154.75 1.00    20073    19912
## disp_disp_group1    28.34      5.82    19.07    41.70 1.00    19072    19408
## disp_disp_group2   193.18     88.33    89.83   419.01 1.00    17579    16115
##
## 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).
## # A tibble: 10 x 5
##    chains  mean    sd     n     se
##     <int> <dbl> <dbl> <int>  <dbl>
##  1      1 -631.  4.46  2000 0.0997
##  2      2 -631.  4.18  2000 0.0935
##  3      3 -631.  4.33  2000 0.0968
##  4      4 -631.  4.42  2000 0.0989
##  5      5 -631.  4.27  2000 0.0956
##  6      6 -631.  4.35  2000 0.0972
##  7      7 -631.  4.38  2000 0.0980
##  8      8 -631.  4.31  2000 0.0963
##  9      9 -631.  4.26  2000 0.0952
## 10     10 -631.  4.24  2000 0.0947
## [1] "Plotting Trace"
## [1] "Plotting violin"
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

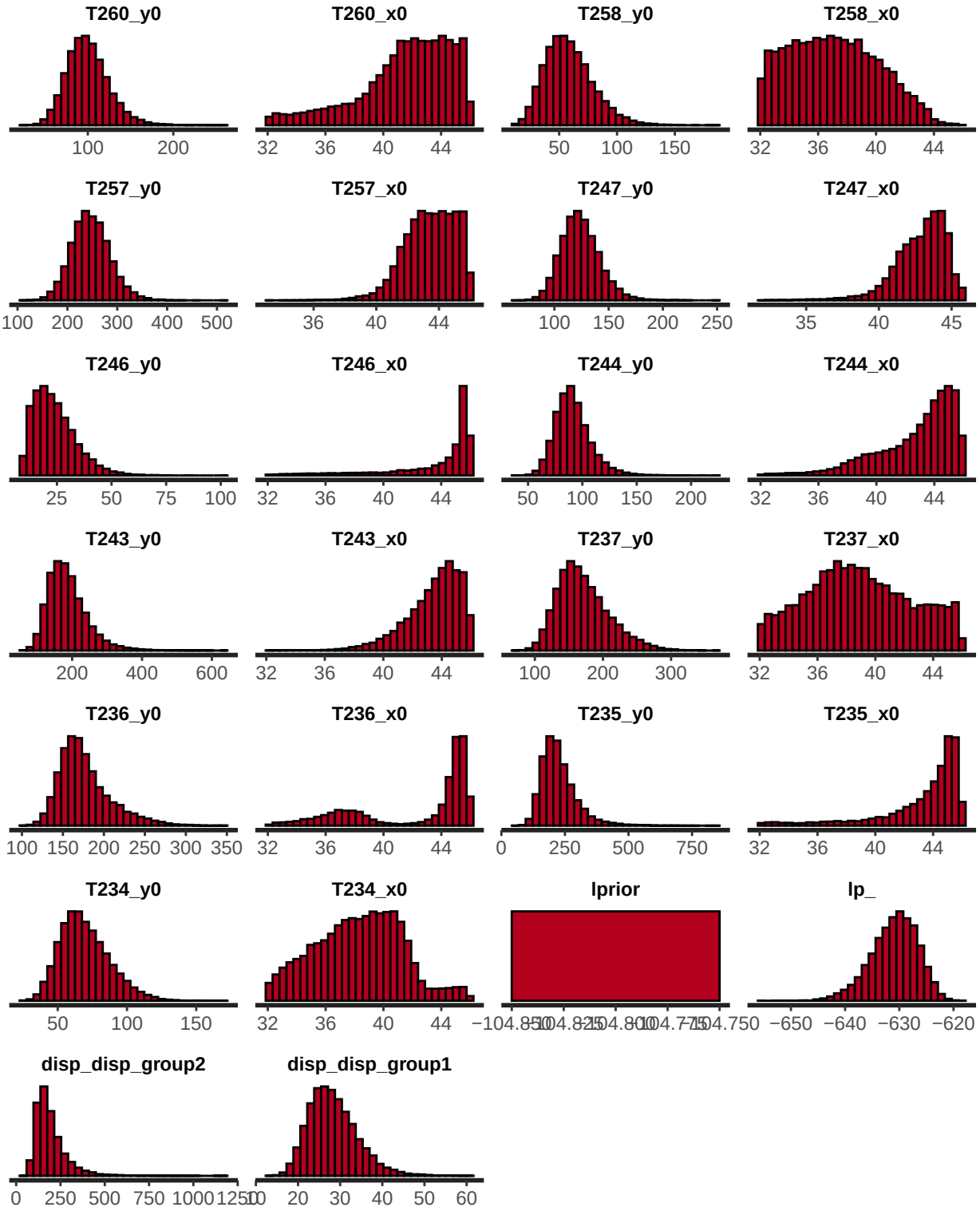



```
## [1] "Plotting hist"
```

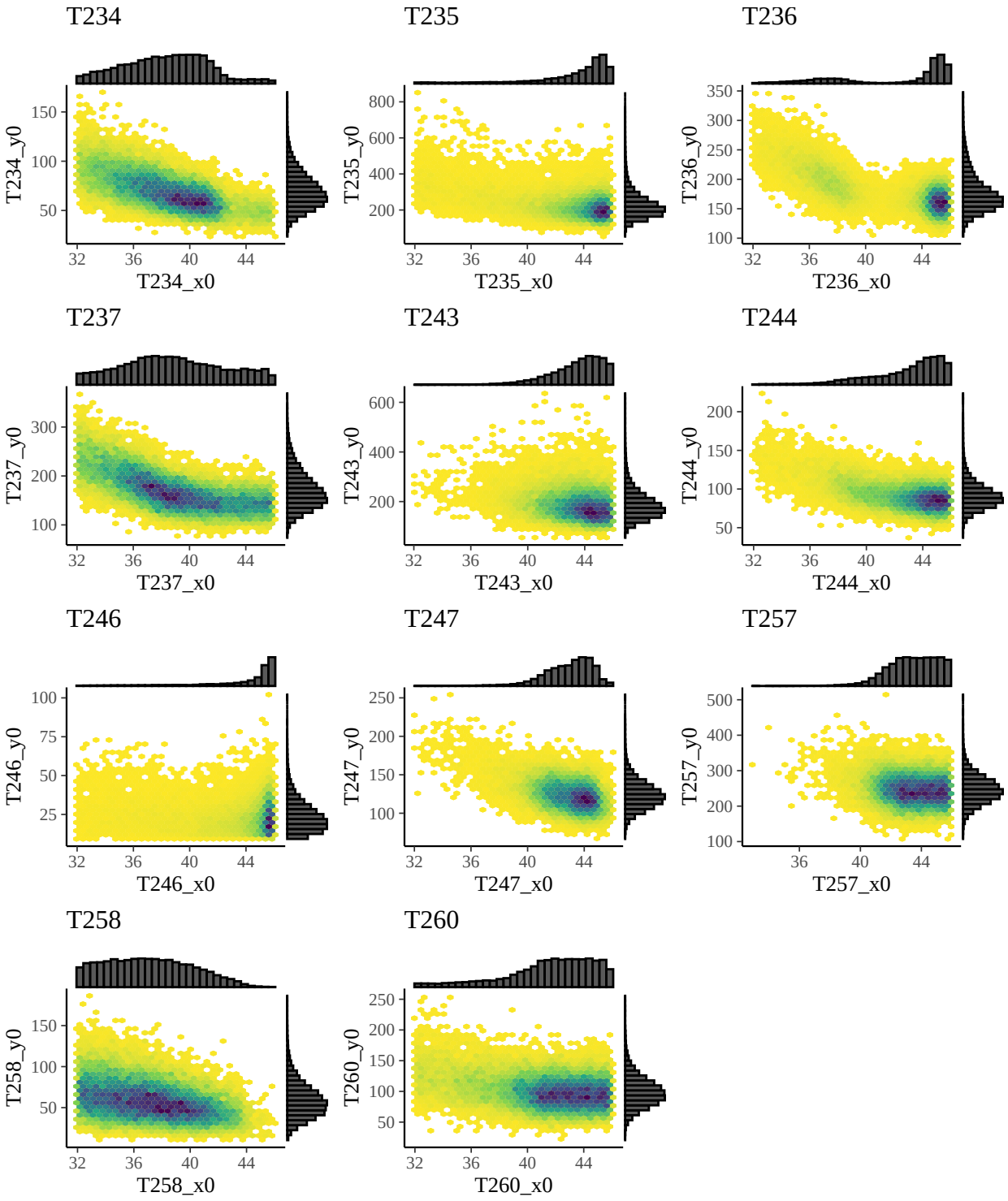



```
## [1] "Plotting Hex"
```

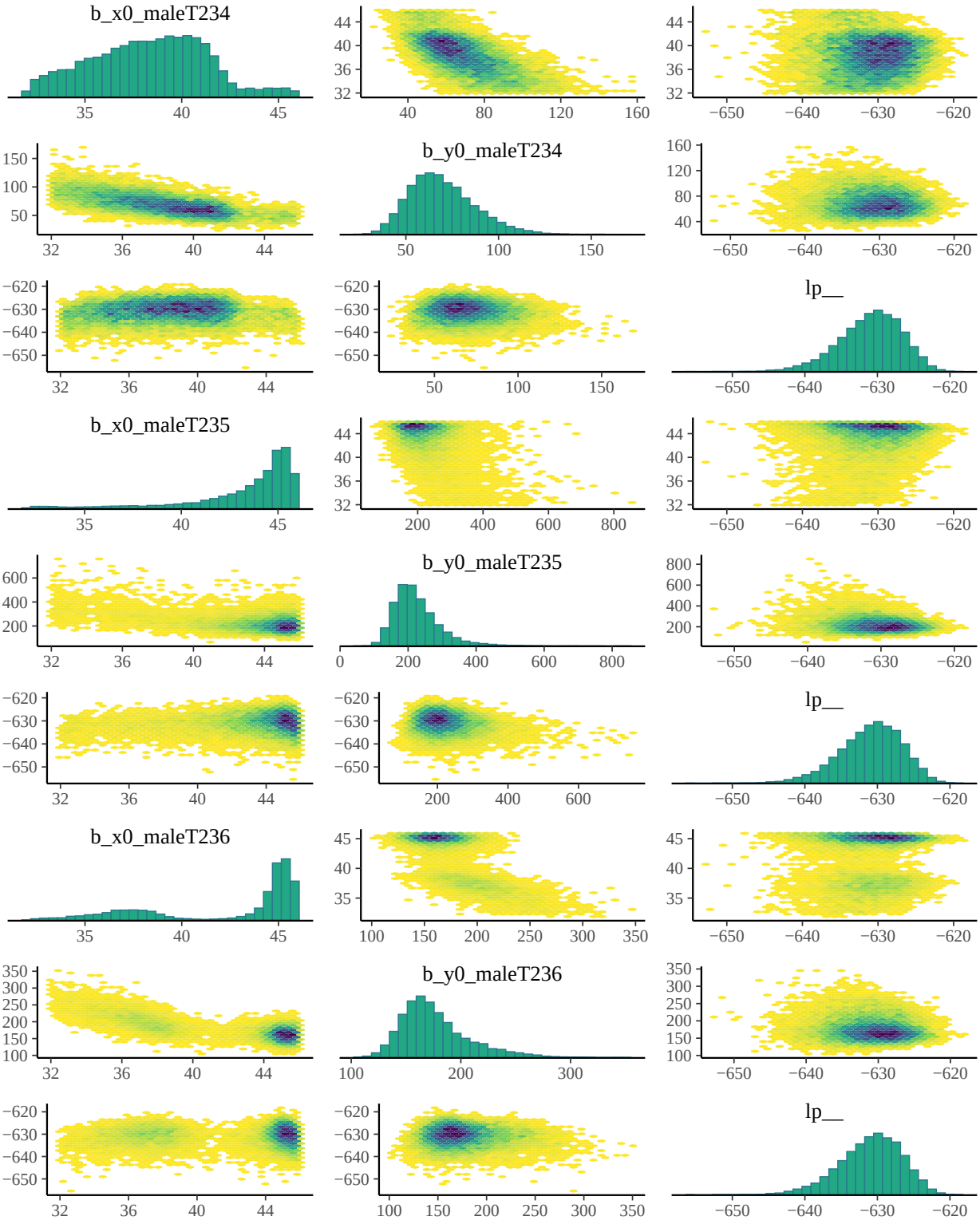
x0: individual; y0: individual; disp_flag: uniform_2; disp prior: 0.01; filter: FALSE

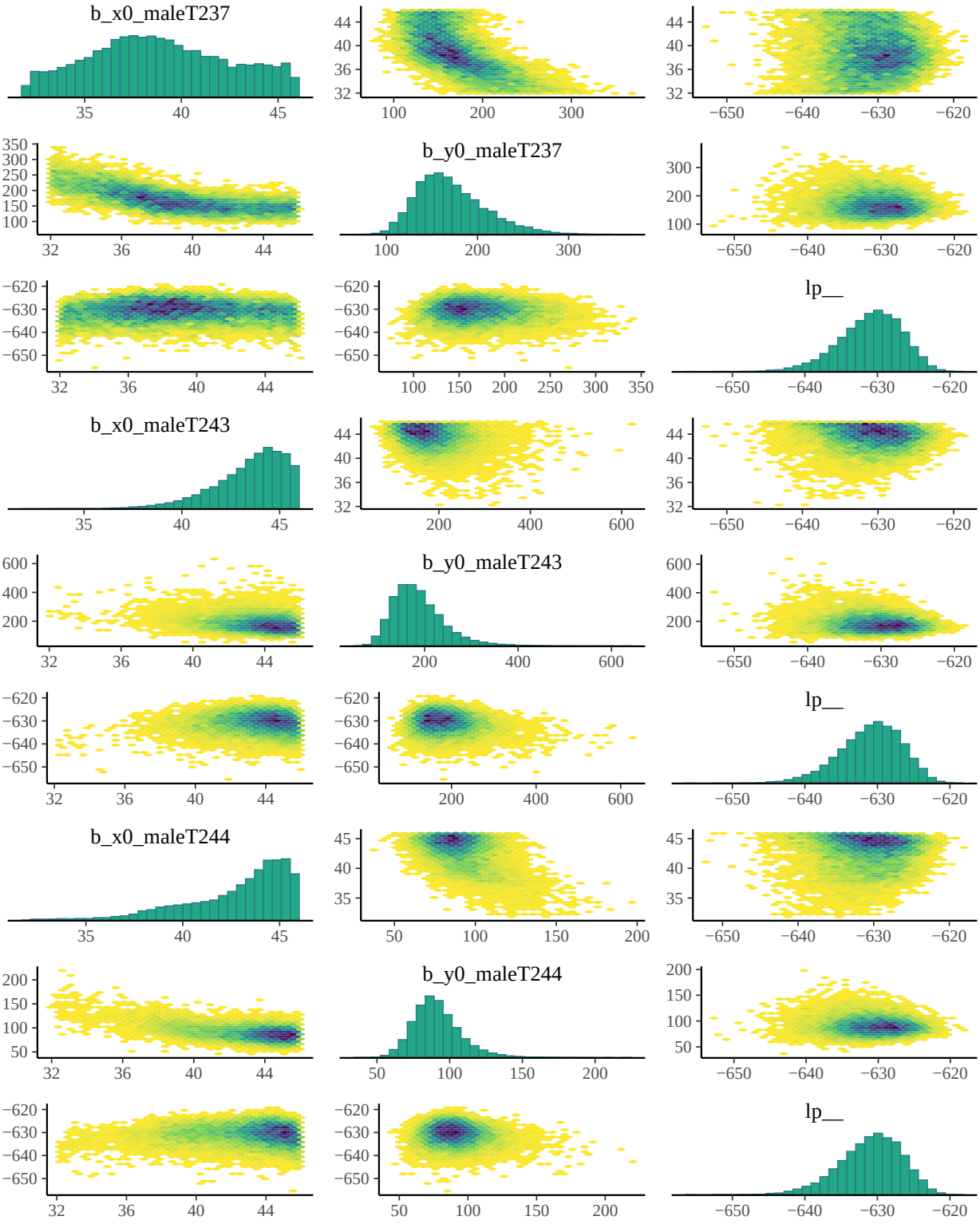


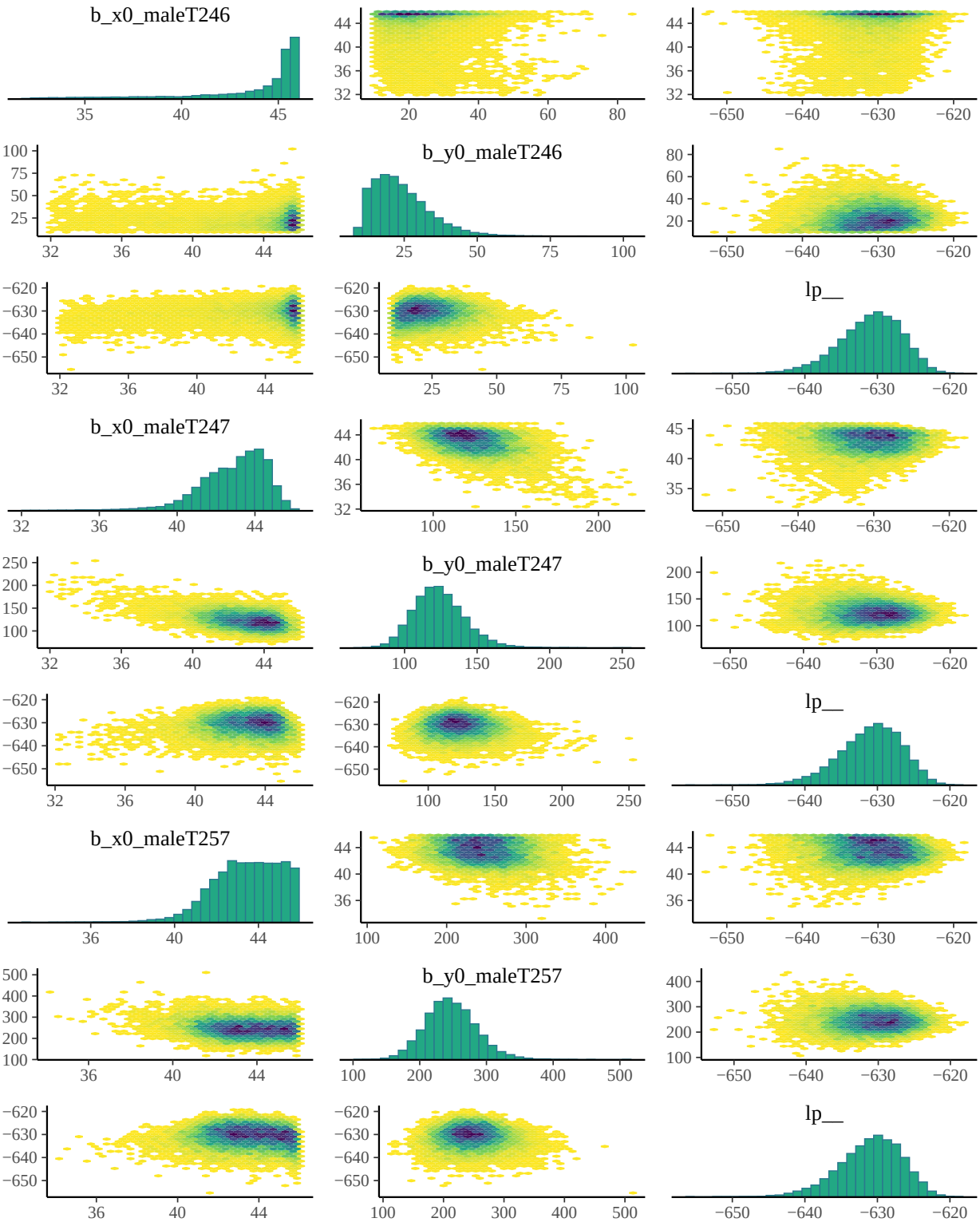
x0: individual; y0: individual; disp_flag: uniform_2; disp prior: 0.01; filter: FALSE




```
## [1] "Plotting Pairs"
```


x0: individual; y0: individual; disp_flag: uniform_2; disp prior: 0.01; filter: FALSE
b_x0_maleT234
b_y0_maleT234
lp__
b_x0_maleT235
b_y0_maleT235
lp__
b_x0_maleT236
b_y0_maleT236
lp__
page 1 of 4


x0: individual; y0: individual; disp_flag: uniform_2; disp prior: 0.01; filter: FALSE
b_x0_maleT237
b_y0_maleT237
lp__
b_x0_maleT243
b_y0_maleT243
lp__
b_x0_maleT244
b_y0_maleT244
lp__

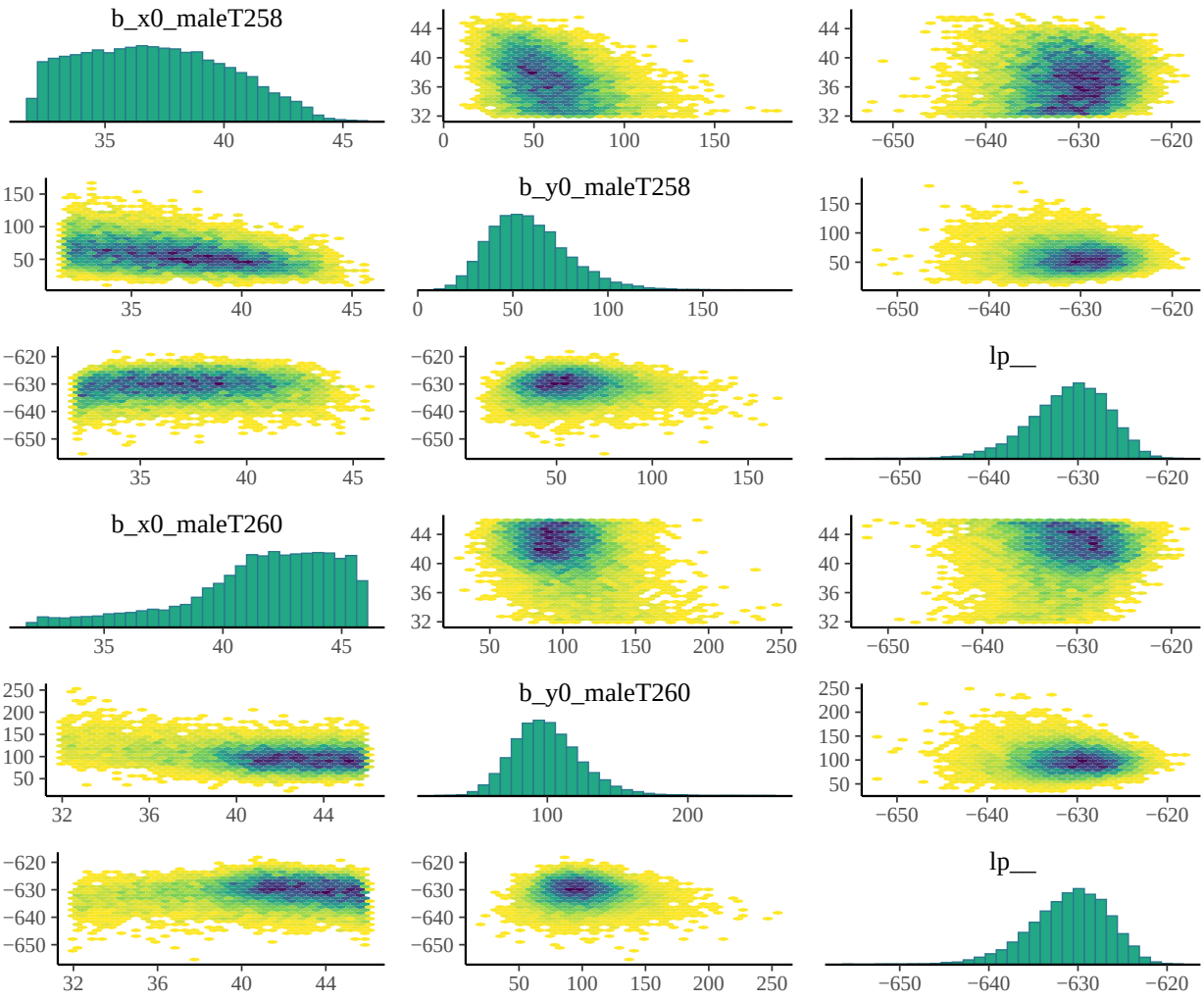


page 3 of 4

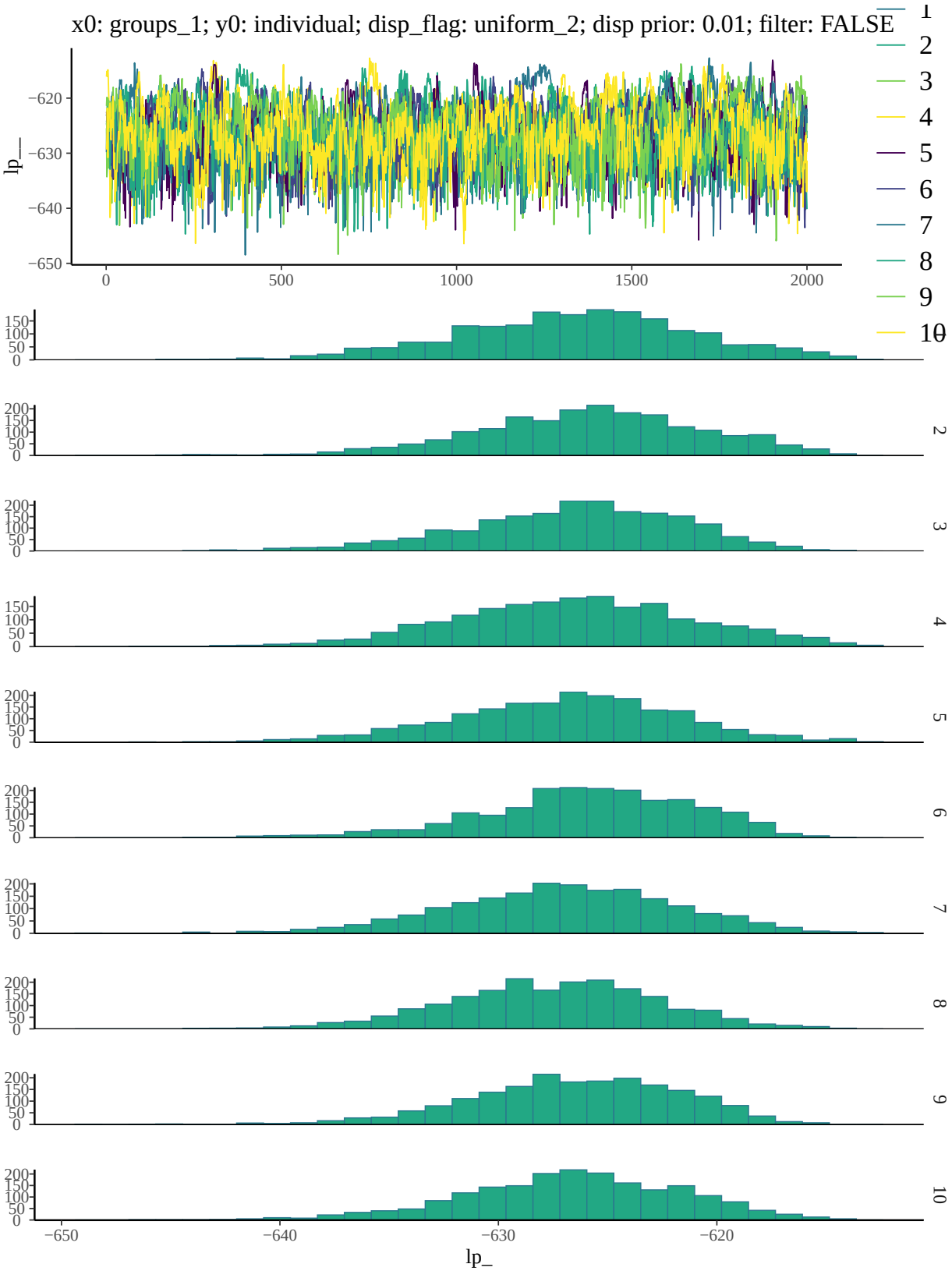
```
## \clearpage[1] "Working with Pre-existing Fits"
## [1] "`class(fit)` = brmsfit"
```

```
##
##
##
## [1] "nbinom_type1; two_piece; x0: groups_1; y0: individual; disp_flag: uniform_2; disp prior: 0.01; i
## [1] "Prior Information"
##                 prior class        coef group resp dpar nlpar lb   ub
##     uniform(32, 45.9)     b                                 x0 32 45.9
##     uniform(32, 45.9)     b   Intercept                     x0 32 45.9
##     uniform(10, 1000)     b                                 y0 10 1000
##     uniform(10, 1000)     b    maleT234                     y0 10 1000
##     uniform(10, 1000)     b    maleT235                     y0 10 1000
##     uniform(10, 1000)     b    maleT236                     y0 10 1000
##     uniform(10, 1000)     b    maleT237                     y0 10 1000
##     uniform(10, 1000)     b    maleT243                     y0 10 1000
##     uniform(10, 1000)     b    maleT244                     y0 10 1000
##     uniform(10, 1000)     b    maleT246                     y0 10 1000
##     uniform(10, 1000)     b    maleT247                     y0 10 1000
##     uniform(10, 1000)     b    maleT257                     y0 10 1000
##     uniform(10, 1000)     b    maleT258                     y0 10 1000
##     uniform(10, 1000)     b    maleT260                     y0 10 1000
##                (flat)     b                         disp
##                (flat)     b disp_group1             disp
##                (flat)     b disp_group2             disp
##  student_t(3, 0, 66.7)    sd                                x0  0   10
##  student_t(3, 0, 66.7)    sd              male              x0  0   10
##  student_t(3, 0, 66.7)    sd   Intercept  male              x0  0   10
##        source
##          user
##  (vectorized)
##          user
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
##      default
##  (vectorized)
##  (vectorized)
##          user
##  (vectorized)
##  (vectorized)
## [1] "nbinom_type1; two_piece; x0: groups_1; y0: individual; disp_flag: uniform_2; disp prior: 0.01; i
## [1] "Fit Information"

## Warning: There were 19703 divergent transitions after warmup. Increasing
## adapt_delta above 0.9 may help. See
## http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup

##  Family: nbinom_type1
```

```
##   Links: mu = identity; disp = identity
## Formula: y ~ two_piece(x, x0, y0)
##          x0 ~ 0 + Intercept + (1 | male)
##          disp ~ 0 + disp_group
##          y0 ~ 0 + male
##    Data: data (Number of observations: 107)
##   Draws: 10 chains, each with iter = 40000; warmup = 30000; thin = 5;
##          total post-warmup draws = 20000
##
## Group-Level Effects:
## ~male (Number of levels: 11)
##                 Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(x0_Intercept)     1.36      1.04     0.06     3.85 1.02      668      754
##
## Population-Level Effects:
##                 Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## x0_Intercept        43.85      1.05    41.46    45.50 1.01      781      782
## y0_maleT234         53.82     11.38    33.99    78.57 1.01      999     1236
## y0_maleT235        213.88     56.35   129.52   350.50 1.01     1286     1846
## y0_maleT236        163.84     19.94   127.70   205.75 1.01     1137     1245
## y0_maleT237        146.19     23.50   104.26   195.79 1.01     1369     1708
## y0_maleT243        181.30     52.95   103.02   311.11 1.01      949      807
## y0_maleT244         88.38     13.96    62.89   118.22 1.01     1262     2027
## y0_maleT246         23.93      9.48    11.03    46.83 1.01     1222     1342
## y0_maleT247        119.80     15.60    91.32   152.19 1.01     1263     1979
## y0_maleT257        244.02     36.87   174.27   320.57 1.01     1337     2216
## y0_maleT258         43.05     15.57    18.09    78.35 1.01     1448     2606
## y0_maleT260         96.24     22.08    56.83   142.53 1.01     1462     2087
## disp_disp_group1    28.39      5.79    19.21    41.63 1.01     1098     2457
## disp_disp_group2   188.48     84.60    88.17   405.68 1.01     1024     1241
##
## 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).
## # A tibble: 10 x 5
##    chains  mean    sd     n    se
##     <int> <dbl> <dbl> <int> <dbl>
##  1      1 -626.  5.40  2000 0.121
##  2      2 -626.  5.12  2000 0.115
##  3      3 -627.  4.95  2000 0.111
##  4      4 -627.  5.46  2000 0.122
##  5      5 -627.  5.12  2000 0.114
##  6      6 -626.  4.81  2000 0.108
##  7      7 -627.  5.13  2000 0.115
##  8      8 -628.  4.79  2000 0.107
##  9      9 -626.  4.73  2000 0.106
## 10     10 -627.  4.95  2000 0.111
## [1] "Plotting Trace"
## [1] "Plotting violin"
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

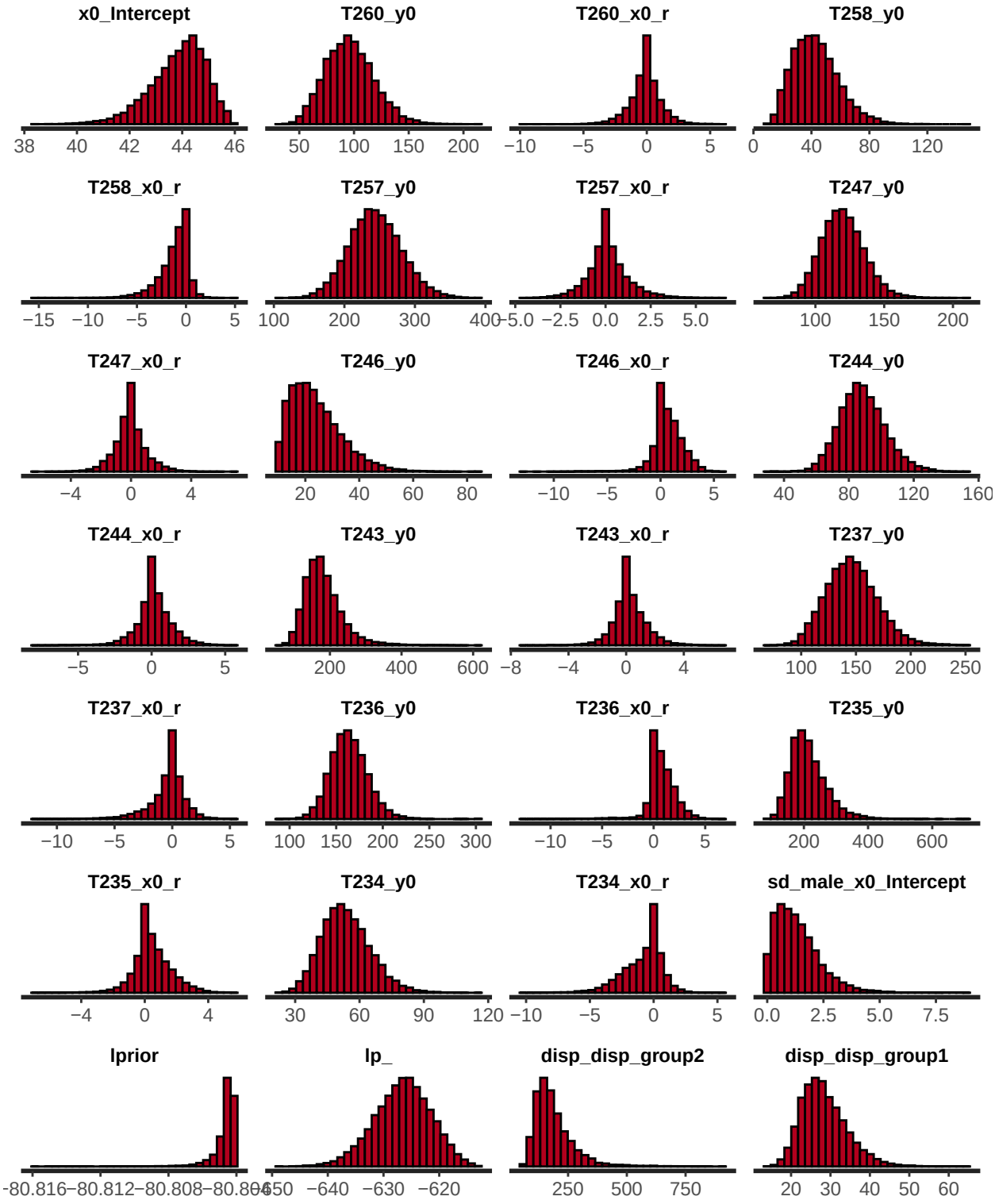



```
## [1] "Plotting hist"
```

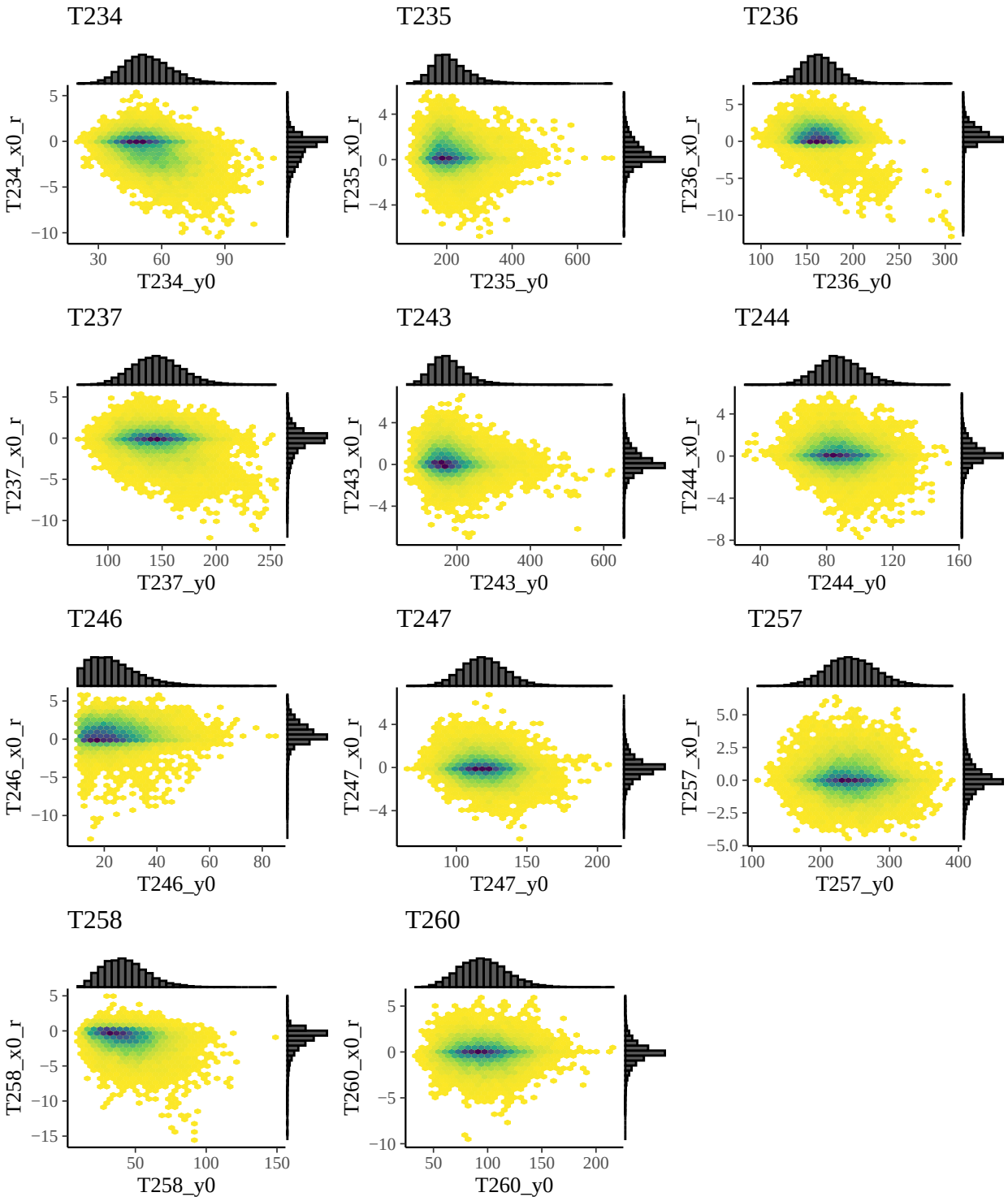



```
## [1] "Plotting Hex"
```

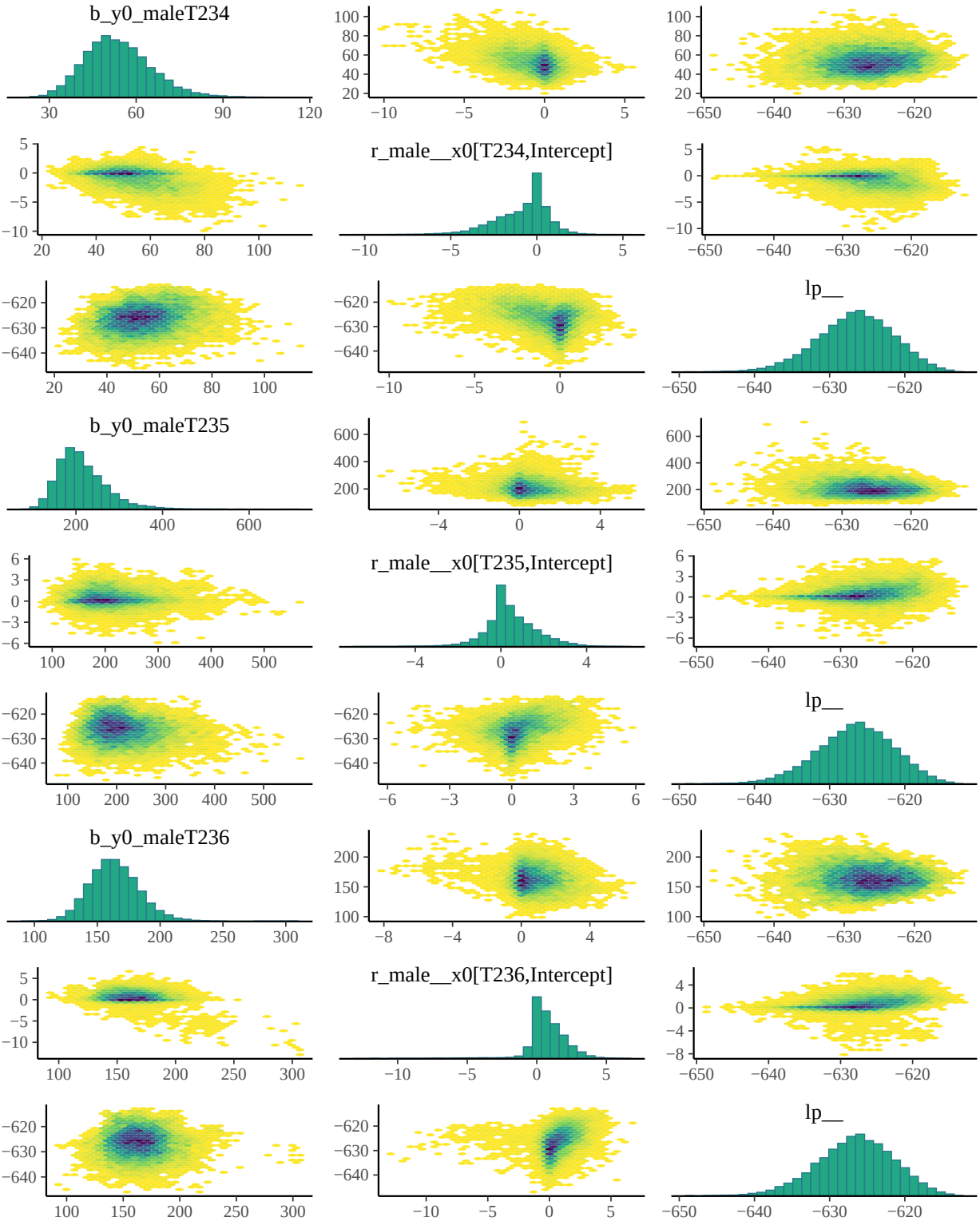
x0: groups_1; y0: individual; disp_flag: uniform_2; disp prior: 0.01; filter: FALSE

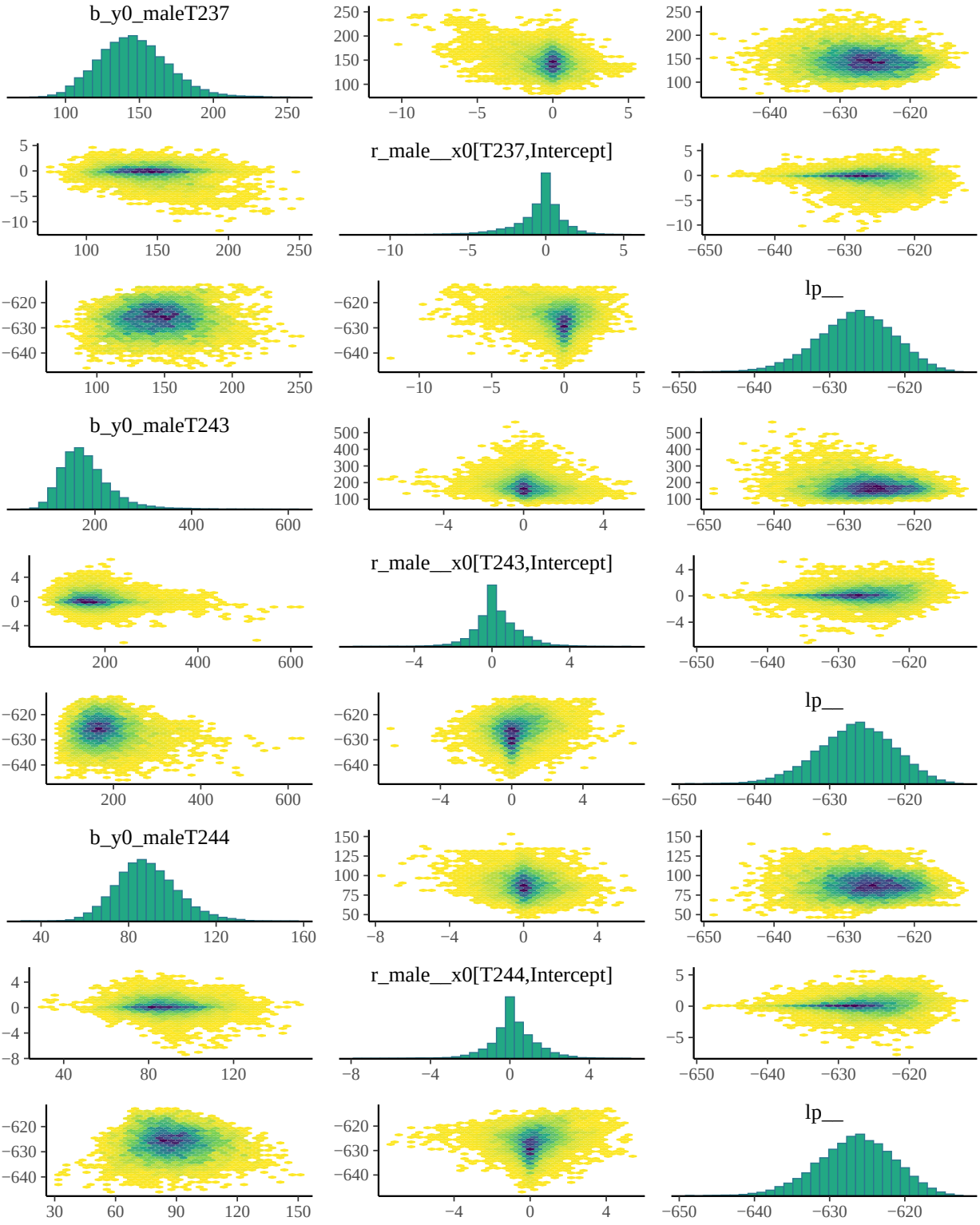


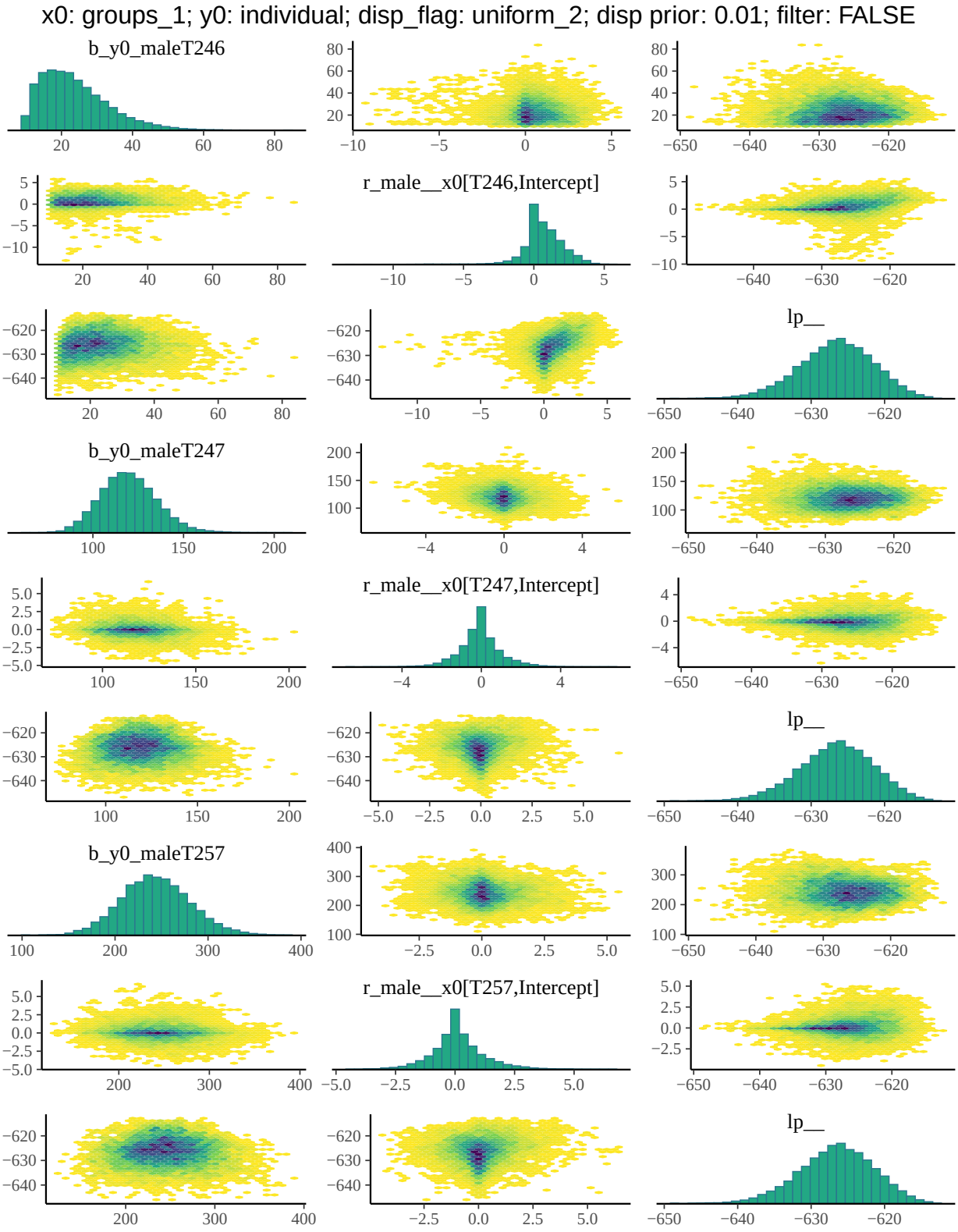
x0: groups_1; y0: individual; disp_flag: uniform_2; disp prior: 0.01; filter: FALSE




```
## [1] "Plotting Pairs"
```


x0: groups_1; y0: individual; disp_flag: uniform_2; disp prior: 0.01; filter: FALSE
b_y0_maleT234
r_male__x0[T234,Intercept]
lp__
b_y0_maleT235
r_male__x0[T235,Intercept]
lp__
b_y0_maleT236
r_male__x0[T236,Intercept]
lp__

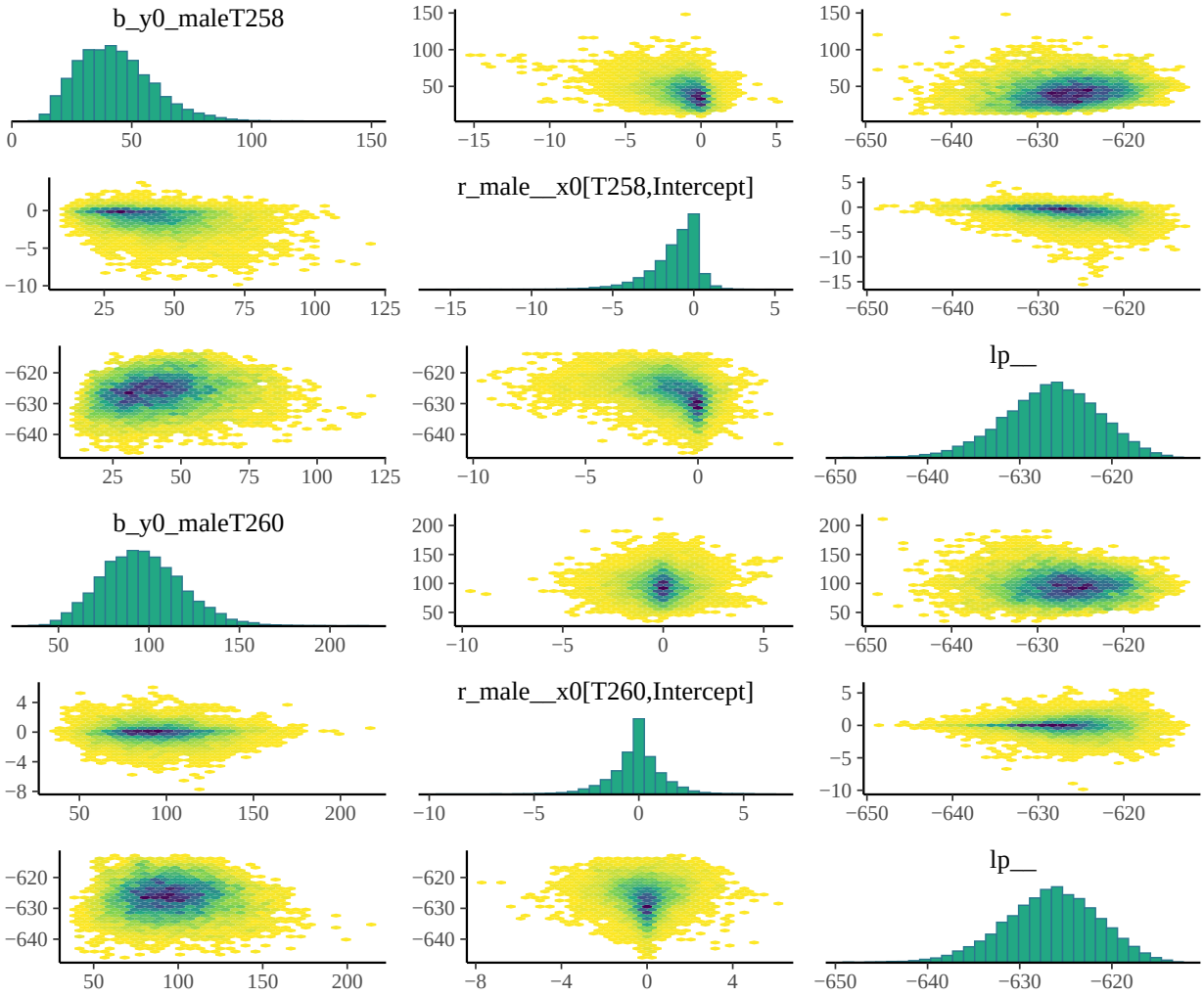

x0: groups_1; y0: individual; disp_flag: uniform_2; disp prior: 0.01; filter: FALSE
b_y0_maleT237
r_male__x0[T237,Intercept]
lp__
b_y0_maleT243
r_male__x0[T243,Intercept]
lp__
b_y0_maleT244
r_male__x0[T244,Intercept]
lp__



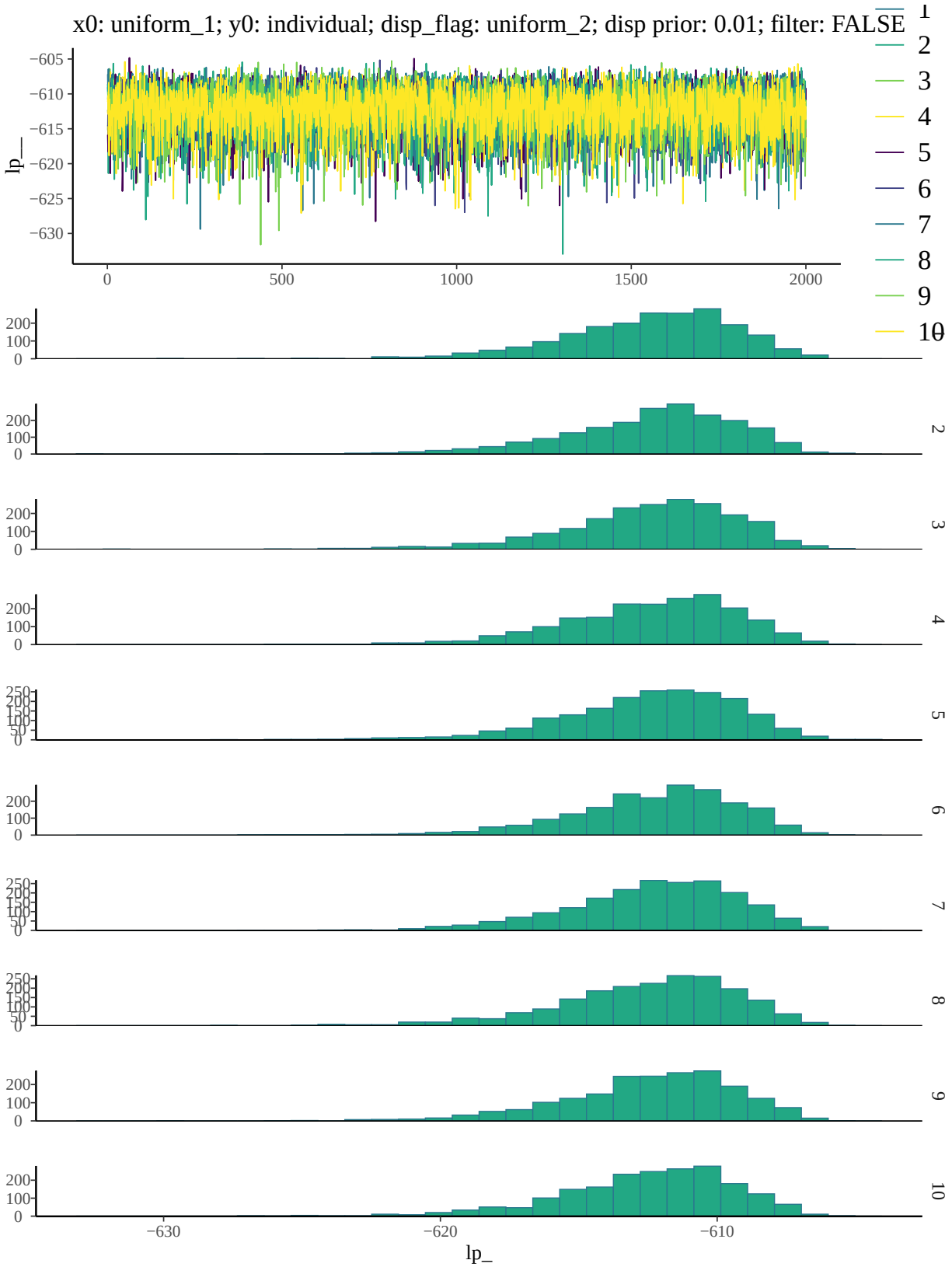

```
## \clearpage[1] "Working with Pre-existing Fits"
## [1] "`class(fit)` = brmsfit"
```

```
##
##
##
## [1] "nbinom_type1; two_piece; x0: uniform_1; y0: individual; disp_flag: uniform_2; disp prior: 0.01;
## [1] "Prior Information"
##              prior class        coef group resp dpar nlpar lb   ub       source
##  uniform(32, 45.9)     b                                 x0 32 45.9         user
##  uniform(32, 45.9)     b   Intercept                     x0 32 45.9 (vectorized)
##  uniform(10, 1000)     b                                 y0 10 1000         user
##  uniform(10, 1000)     b    maleT234                     y0 10 1000 (vectorized)
##  uniform(10, 1000)     b    maleT235                     y0 10 1000 (vectorized)
##  uniform(10, 1000)     b    maleT236                     y0 10 1000 (vectorized)
##  uniform(10, 1000)     b    maleT237                     y0 10 1000 (vectorized)
##  uniform(10, 1000)     b    maleT243                     y0 10 1000 (vectorized)
##  uniform(10, 1000)     b    maleT244                     y0 10 1000 (vectorized)
##  uniform(10, 1000)     b    maleT246                     y0 10 1000 (vectorized)
##  uniform(10, 1000)     b    maleT247                     y0 10 1000 (vectorized)
##  uniform(10, 1000)     b    maleT257                     y0 10 1000 (vectorized)
##  uniform(10, 1000)     b    maleT258                     y0 10 1000 (vectorized)
##  uniform(10, 1000)     b    maleT260                     y0 10 1000 (vectorized)
##             (flat)     b                        disp                    default
##             (flat)     b disp_group1            disp                (vectorized)
##             (flat)     b disp_group2            disp                (vectorized)
## [1] "nbinom_type1; two_piece; x0: uniform_1; y0: individual; disp_flag: uniform_2; disp prior: 0.01;
## [1] "Fit Information"
##  Family: nbinom_type1
##   Links: mu = identity; disp = identity
## Formula: y ~ two_piece(x, x0, y0)
##          x0 ~ 0 + Intercept
##          disp ~ 0 + disp_group
##          y0 ~ 0 + male
##    Data: data (Number of observations: 107)
##   Draws: 10 chains, each with iter = 40000; warmup = 30000; thin = 5;
##          total post-warmup draws = 20000
##
## Population-Level Effects:
##                 Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## x0_Intercept       44.90      0.75    43.23    45.87 1.00    18701    18410
## y0_maleT234        50.37     10.04    32.78    71.80 1.00    19658    18888
## y0_maleT235       212.37     56.63   126.99   345.46 1.00    18379    17823
## y0_maleT236       163.66     19.26   127.93   203.17 1.00    19938    19517
## y0_maleT237       144.72     22.65   102.83   191.55 1.00    19362    19324
## y0_maleT243       172.86     49.86    97.37   293.50 1.00    18886    17389
## y0_maleT244        87.24     13.55    62.39   115.63 1.00    19577    19680
## y0_maleT246        24.16      9.68    10.87    47.30 1.00    18787    17856
## y0_maleT247       116.66     15.94    87.16   149.73 1.00    19752    19706
## y0_maleT257       243.82     37.36   174.41   321.78 1.00    19560    19582
## y0_maleT258        34.98     12.94    14.33    64.47 1.00    17823    16885
## y0_maleT260        97.43     22.77    56.72   145.47 1.00    19955    19191
## disp_disp_group1   30.10      6.06    20.52    44.41 1.00    20004    19359
## disp_disp_group2  185.49     84.81    86.13   401.65 1.00    17466    15885
##
## 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).
## # A tibble: 10 x 5
##    chains  mean    sd     n     se
##     <int> <dbl> <dbl> <int>  <dbl>
##  1      1 -612.  3.04  2000 0.0681
##  2      2 -612.  3.16  2000 0.0706
##  3      3 -612.  3.14  2000 0.0703
##  4      4 -612.  3.07  2000 0.0686
##  5      5 -612.  3.12  2000 0.0699
##  6      6 -612.  3.01  2000 0.0673
##  7      7 -612.  3.01  2000 0.0672
##  8      8 -612.  3.17  2000 0.0709
##  9      9 -612.  3.07  2000 0.0688
## 10     10 -612.  3.09  2000 0.0691
## [1] "Plotting Trace"
## [1] "Plotting violin"
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

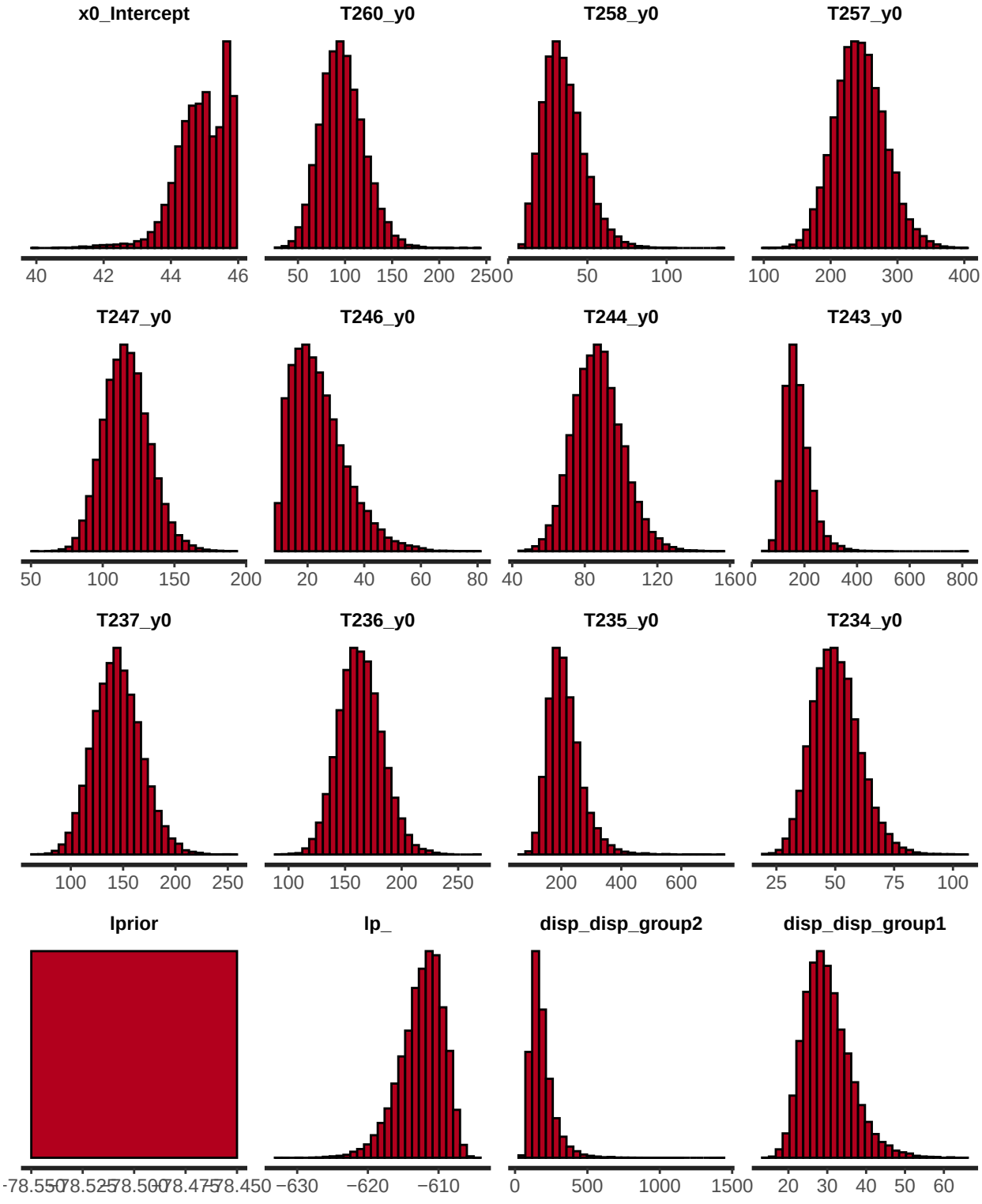



```
## [1] "Plotting hist"
```

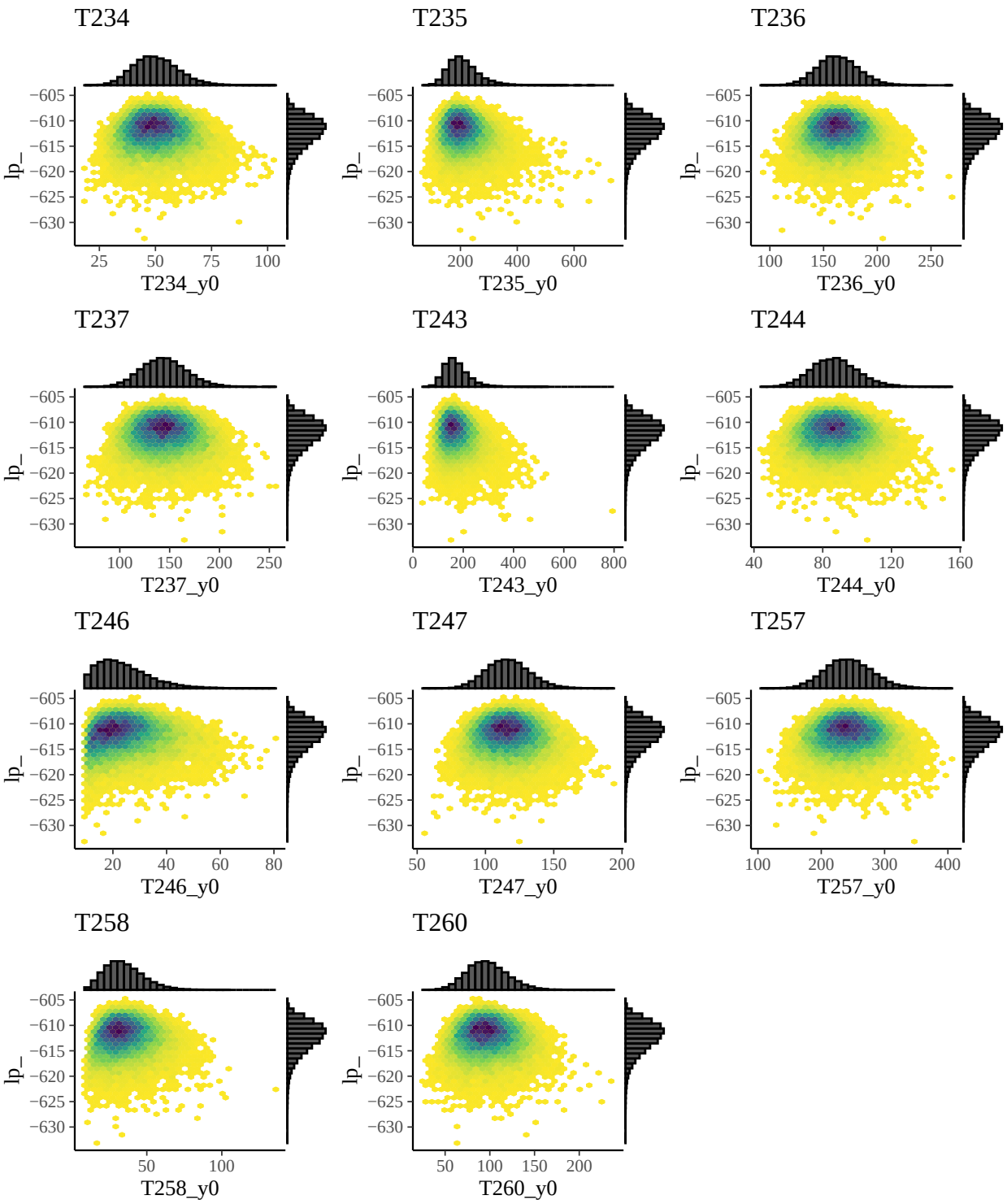



```
## [1] "Plotting Hex"
```

x0: uniform_1; y0: individual; disp_flag: uniform_2; disp prior: 0.01; filter: FALSE

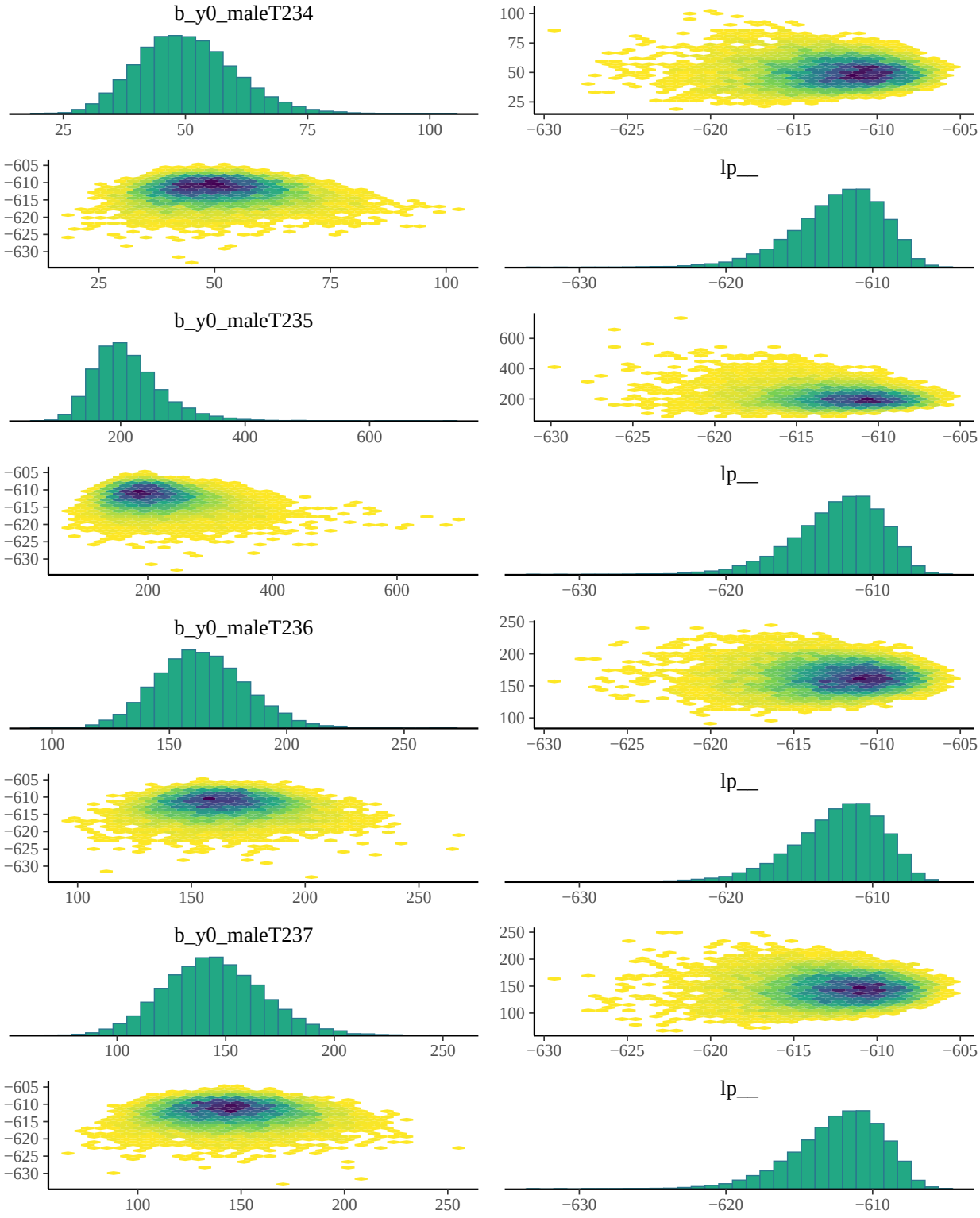


x0: uniform_1; y0: individual; disp_flag: uniform_2; disp prior: 0.01; filter: FALSE

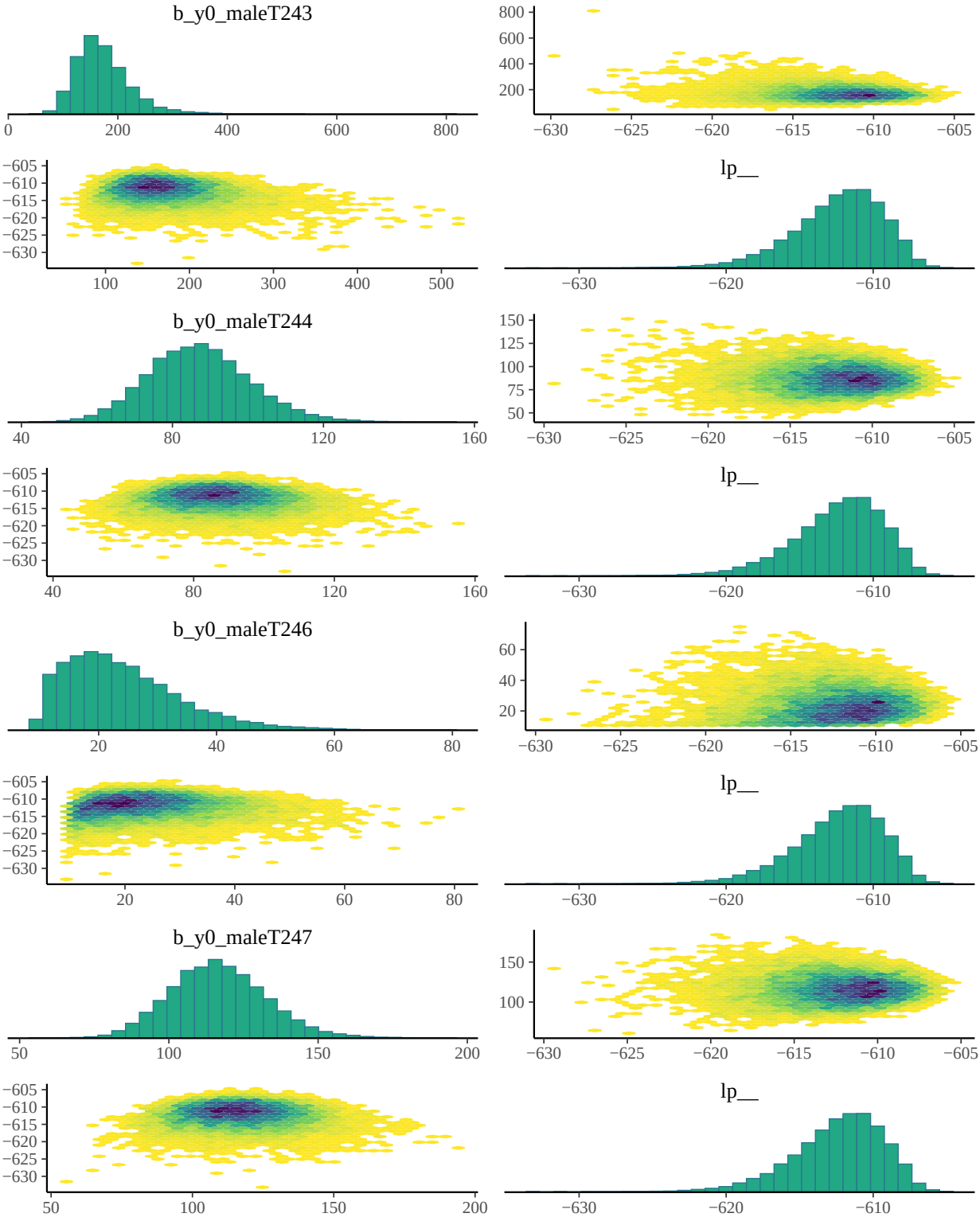


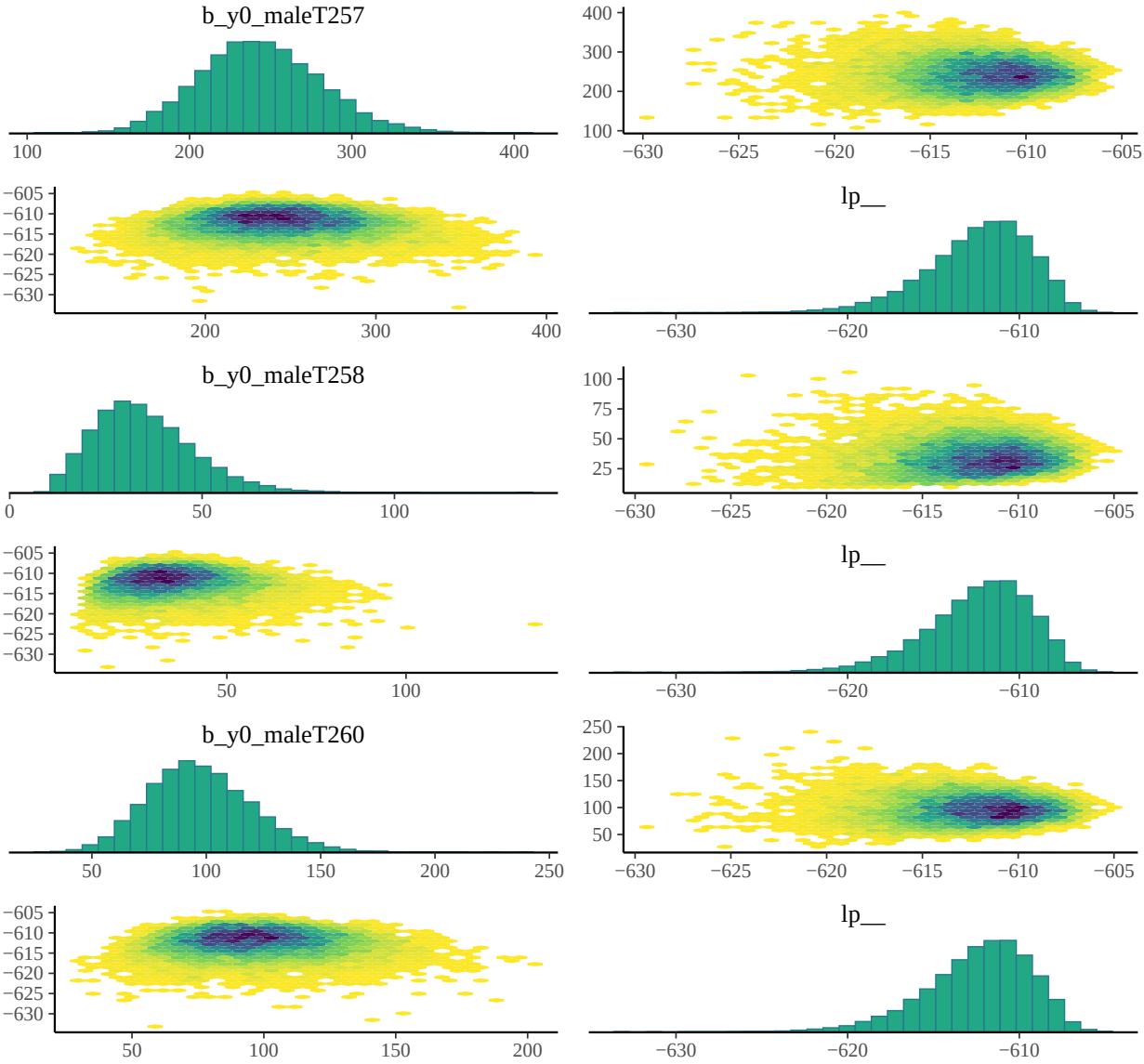

```
## [1] "Plotting Pairs"
```

x0: uniform_1; y0: individual; disp_flag: uniform_2; disp prior: 0.01; filter: FALSE



x0: uniform_1; y0: individual; disp_flag: uniform_2; disp prior: 0.01; filter: FALSE




x0: uniform_1; y0: individual; disp_flag: uniform_2; disp prior: 0.01; filter: FALSE
b_y0_maleT257
lp__
b_y0_maleT258
lp__
b_y0_maleT260
lp__


```
## \clearpage
```

```
# rm("curr_row")
```

```
print(outfile_tbl)
```

```
## [1] NA
```

```
if(save_fits) save(fit_tbl, file = outfile_tbl)
```

Exit rendering

```
knitr::knit_exit()
```