Supplementary material - Orexin Cell transplant reduces behavioral arrest severity in narcoleptic mice.

Ana Clementina Equihua-Benítez, Julián A. Equihua-Benítez, Khalil Guzmán-Vásquez, Oscar Prospero-García & René Drucker-Colín.

Introduction

The differences in groups was analyzed using Bayesian multilevel models. These analyses were carried out using the R language (Team, 2018) by means of the brms package (Burkner, 2017) which is a high level interface to the Stan probabilistic programming language (Carpenter et al.,2017).

A total of six models were fitted for this purpose. Three (1-3) to analyze the treatment effects on BAs, and one model (4-6) to analyze the treatment effects on sleep architecture.

All models showed a good convergence and fit.

1. A model fit to predict BA duration, with response distribution set to log-normal, with the variable group as the population-level parameter of interest and a varying-subject intercept, following the terminology of (Bürkner & Vuorre, 2019) .
2. A model to predict the total number of BAs per sampled hour as the total sampled time varied slightly for each subject. Response distribution is set to Poisson, also with the variable group as the population-level parameter of interest. But as what is modeled are rates, we introduced a log(hours) offset.
3. A model to predict the percentage of total time spent in a BA state. As this response is limited to 0 and 1, it was set to be a Beta distribution. Again, the variable group is the population-level parameter of interest.
4. A model to predict the total time in seconds spent in a REM sleep. With response distribution set to log-normal, with the variable group as the population-level parameter of interest.
5. A model to predict the total time in seconds spent in a NREM sleep. With response distribution set to log-normal, with the variable group as the population-level parameter of interest.
6. A model to predict the total time in seconds spent awake. With response distribution set to log-normal, with the variable group as the population-level parameter of interest.

```
# Load needed packages.

library("tidyverse")
library("ggplot2")
library("brms")
library("brmstools")
library("bayesplot")
library("emmeans")
library("tidybayes")
library("here")
library("ggmcmc")
```

Treatment effects on BAs

Models 1-3 are fit on the data generated by manually labelling mouse video recordings.

We read in the data and munge it to have it in the necessary format to fit each model.

BA duration model

```
# Load misc. functions.
source(here("R", "misc_functions.R"))

# Load recorded behaviors data.
data <- read_csv(here("data","videos","recorded_behaviors.csv"))

# Select only BA events (units are seconds).
ba_seconds <- filter(data,behavior=="BA")

# Change variable types.
ba_seconds$group <- as.factor(ba_seconds$group)
ba_seconds$mouse <- as.factor(ba_seconds$mouse)
```

The default settings in the brms package for priors were used in all models. For the fitted models each group-level effect of the grouping factor has a standard deviation. By default ordinary regression coefficients have improper flat priors and the rest are restricted to be non-negative and have a half Student's t-distribution prior (Gelman, 2006) with 3 degrees of freedom and a scale parameter that depends on the standard deviation of the response after applying the link function (Bürkner, 2017). These are equivalent to having no priors and weakly informative priors respectively. For example, for the BA duration model, 9 parameters need to be estimated.

```
# Which priors can we specify for our model?
get_prior(seconds ~ group + (1|mouse), data=ba_seconds)
```

```
##                  prior     class      coef group resp dpar nlpar bound
## 1                              b
## 2                              b groupsham
## 3                              b grouptxcb
## 4                              b grouptxox
## 5 student_t(3, 26, 18) Intercept
## 6  student_t(3, 0, 18)        sd
## 7                              sd           mouse
## 8                              sd Intercept mouse
## 9  student_t(3, 0, 18)     sigma
```

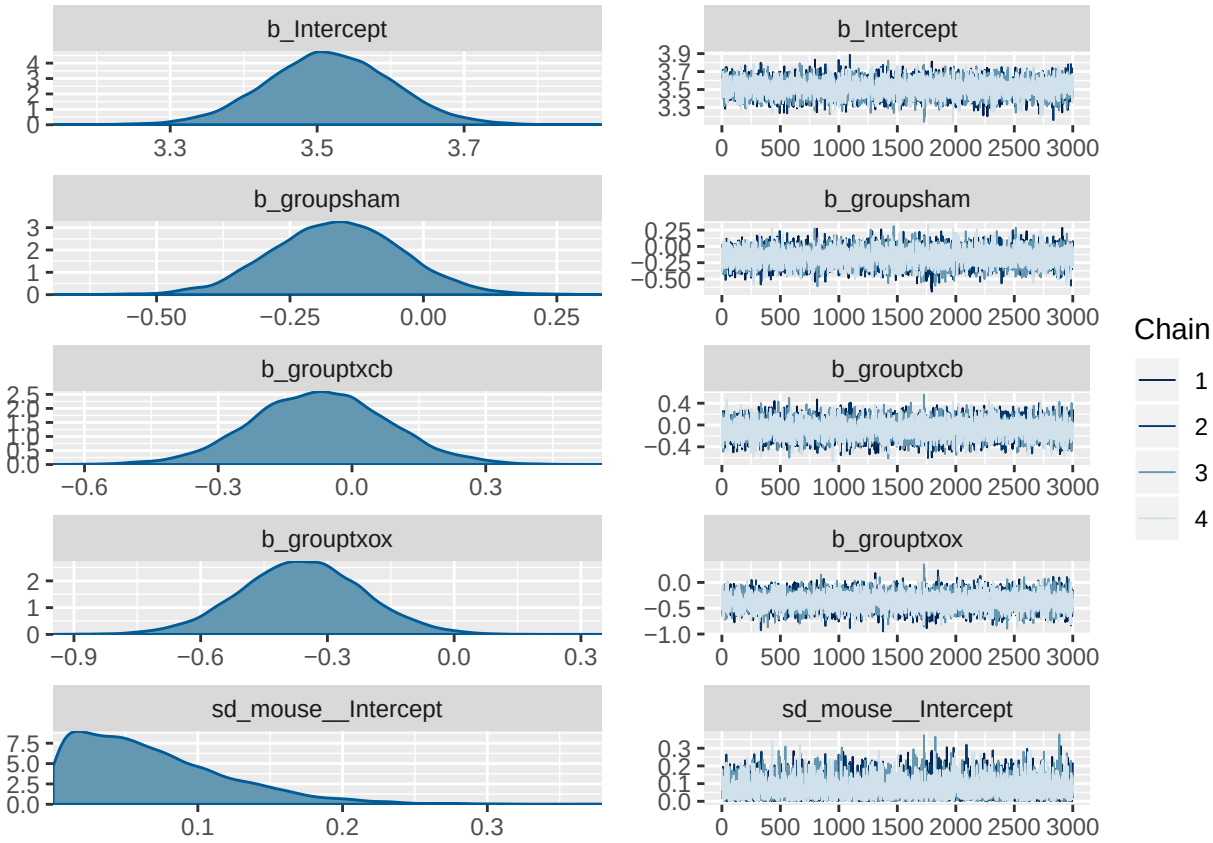
We proceed to fit the model.

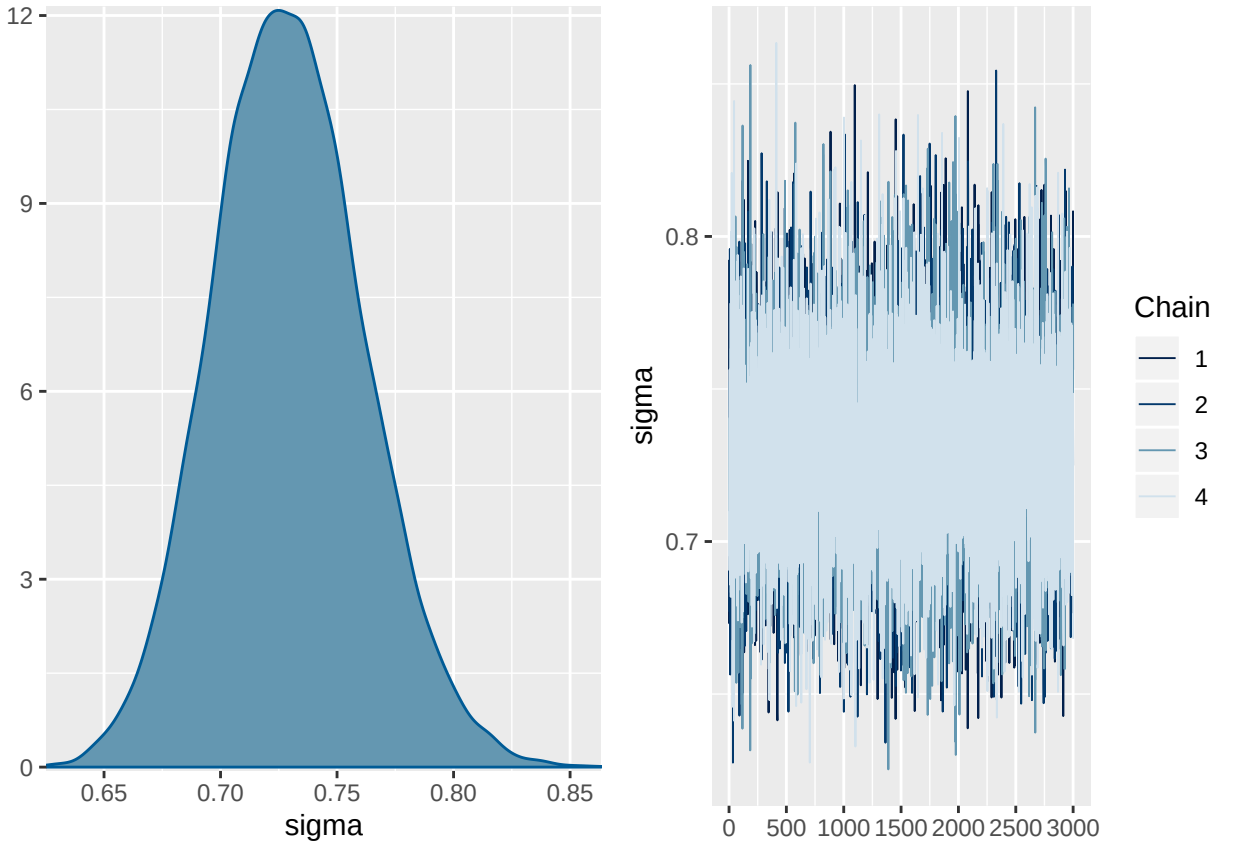
```
# Fit brms model for duration of each BA attack.
ba_seconds_fit <- brm(seconds ~ group + (1|mouse),
                      data = ba_seconds,
                      family = lognormal,
                      warmup  = 2000,
                      iter = 5000,
                      control = list(adapt_delta = 0.999))
```

No runtime warnings during estimation were thrown which is a first indication of proper convergence.

We perform a couple of more checks. First we see if the trace plot exhibits convergence.

```
plot(ba_seconds_fit)
```

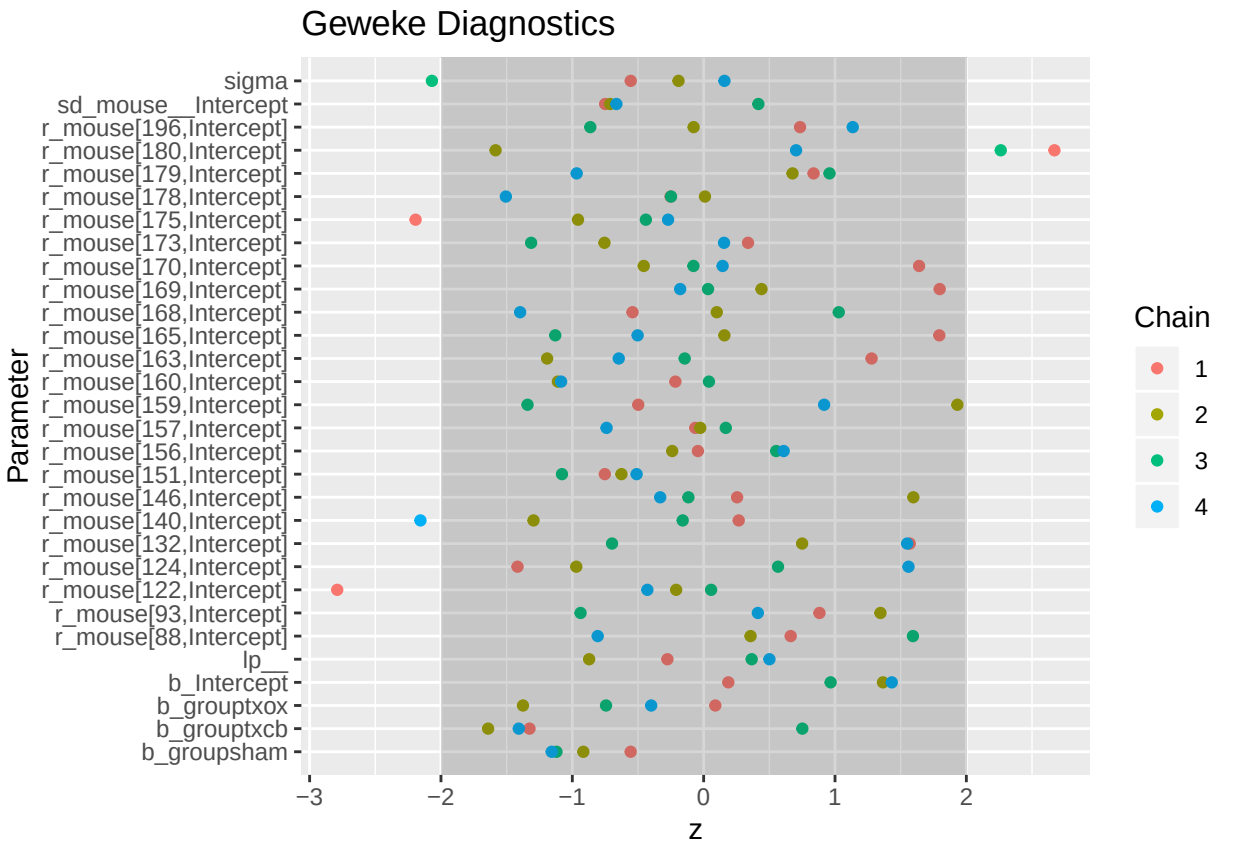




The classical "furry caterpillars" can be seen which is an indication of no divergences.

We then check the Geweke Diagnostic (Geweke, 1992) which shows the Z-scores for a test of equality of means between the first and last parts of each chain (usually the first 10% and the last 50%). These should be in the interval [-1.96,1.96]. Scores above 1.96 or below -1.96 indicate that the two portions of the chain differ significantly and full chain convergence was not obtained (Lesaffre & Lawson, 2012).

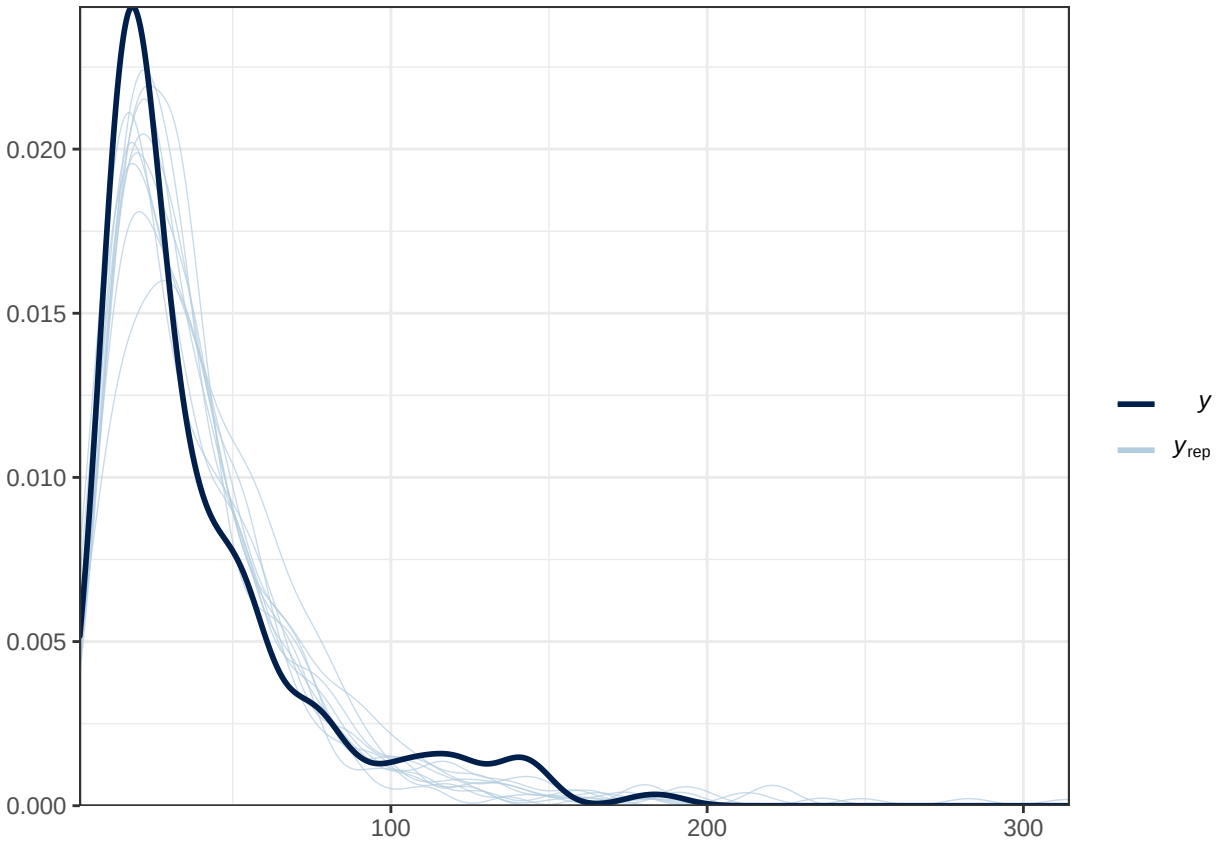
```
mcmc_df = ggs(as.mcmc(ba_seconds_fit))

ggs_geweke(mcmc_df,frac1 = 0.1, frac2 = 0.5)
```



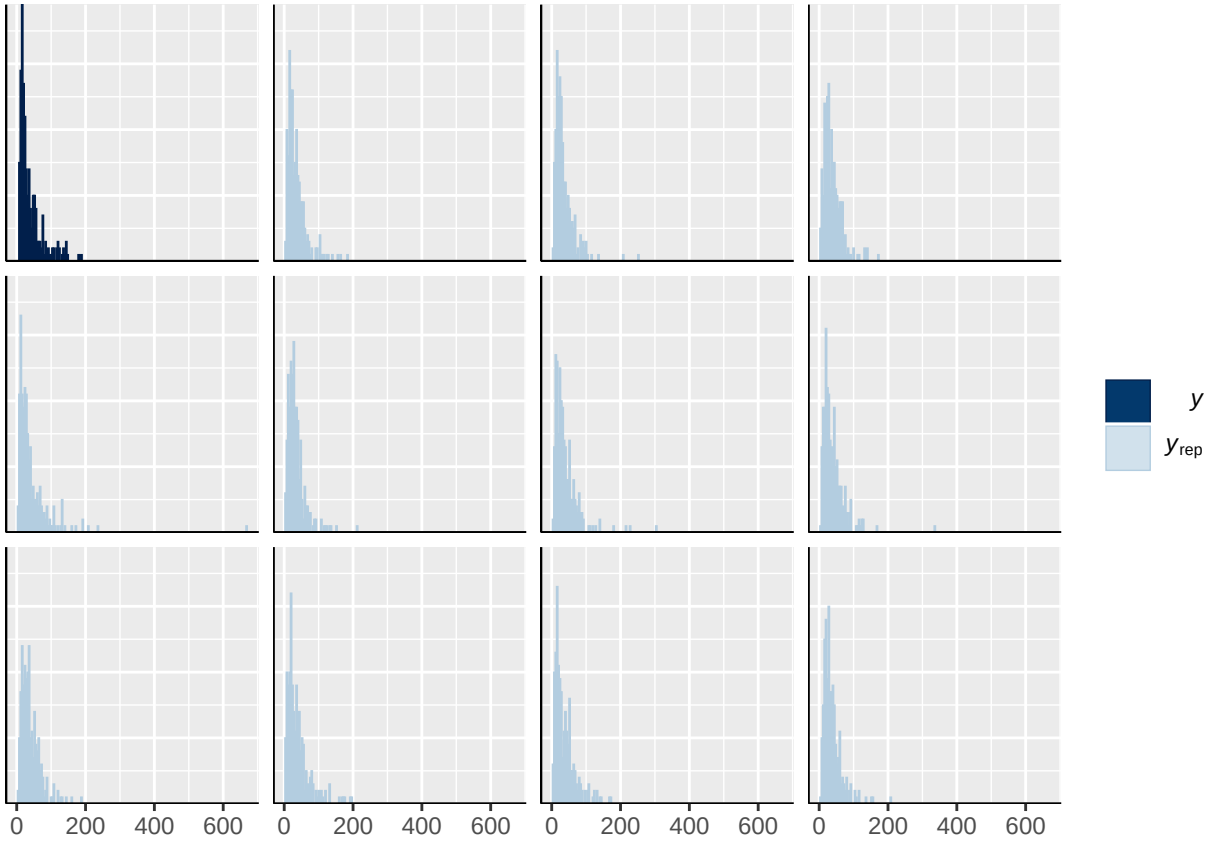
Except for a small amount, the Z-values are within [-1.96,1.96], meaning that the burn-in and the total chain iterations were sufficient for convergence.

Since we now have a data-informed data generating model, we can assess model fit by examining how well the model can reproduce (simulate) the data at hand. This is achieved by examining (visual) Posterior Predictive (PP) checks, which are vital for bayesian model evaluation (Gabry et al., 2019).

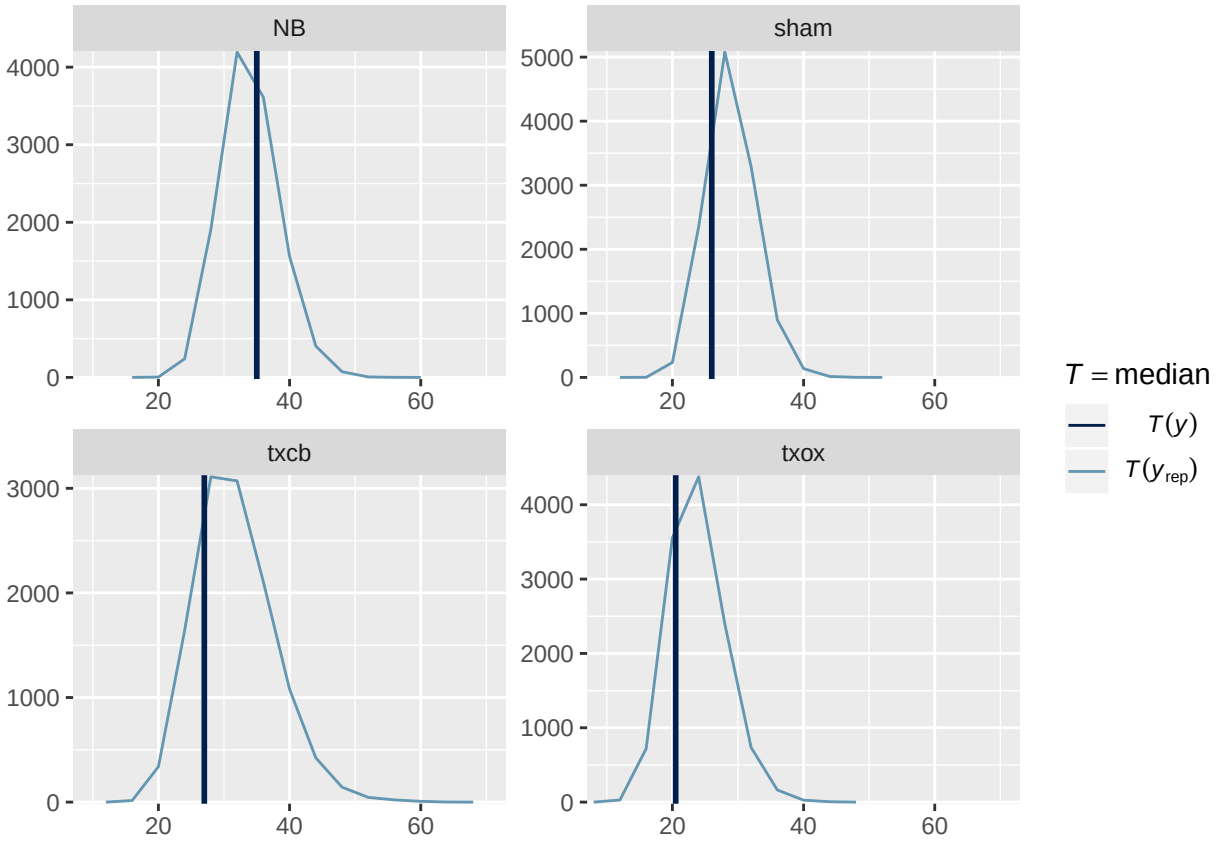
```
# Posterior Predictive checks.

# PP kernel density plot (observed vs simulated values).
pp <- brms::pp_check(ba_seconds_fit,nsamples=10)
pp + theme_bw()
```




```
# PP histogram plot (observed vs simulated values).
y_simulated <- posterior_predict(ba_seconds_fit, draws = 1000)
ppc_hist(ba_seconds$seconds, y_simulated[1:11,], binwidth = 4)
```




```
# PP frecuency polygons vs the observed median of each group.
ppc_stat_freqpoly_grouped(ba_seconds$seconds, y_simulated,
                          group=ba_seconds$group, stat = "median",binwidth = 4)+
                          coord_cartesian(xlim = c(10, 70))
```



Fit seems adequate as densities and histograms of observed and predicted values are similar.

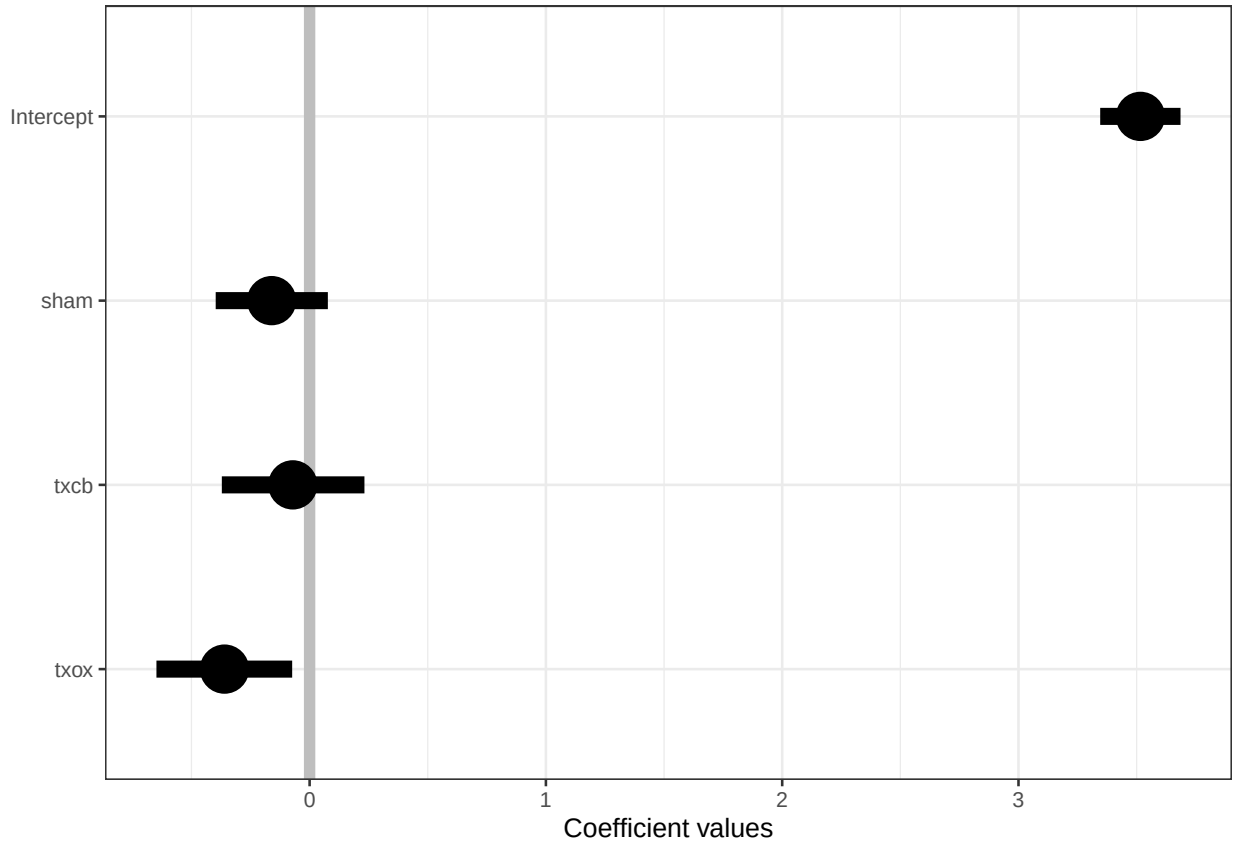
Finally, we interpret our model results.

```
# Model summary.
summary(ba_seconds_fit)
```

```
##  Family: lognormal
##   Links: mu = identity; sigma = identity
## Formula: seconds ~ group + (1 | mouse)
##    Data: ba_seconds (Number of observations: 258)
## Samples: 4 chains, each with iter = 5000; warmup = 2000; thin = 1;
##          total post-warmup samples = 12000
##
## Group-Level Effects:
## ~mouse (Number of levels: 23)
##               Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
## sd(Intercept)     0.07      0.05     0.00     0.20       5627 1.00
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
## Intercept     3.52      0.09     3.35     3.69       7705 1.00
## groupsham    -0.16      0.12    -0.40     0.08       8046 1.00
## grouptxcb    -0.07      0.15    -0.37     0.23       9878 1.00
## grouptxox    -0.36      0.15    -0.65    -0.07       9682 1.00
##
```

```
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
## sigma     0.73      0.03     0.67     0.80      15081 1.00
##
## Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
## is a crude measure of effective sample size, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

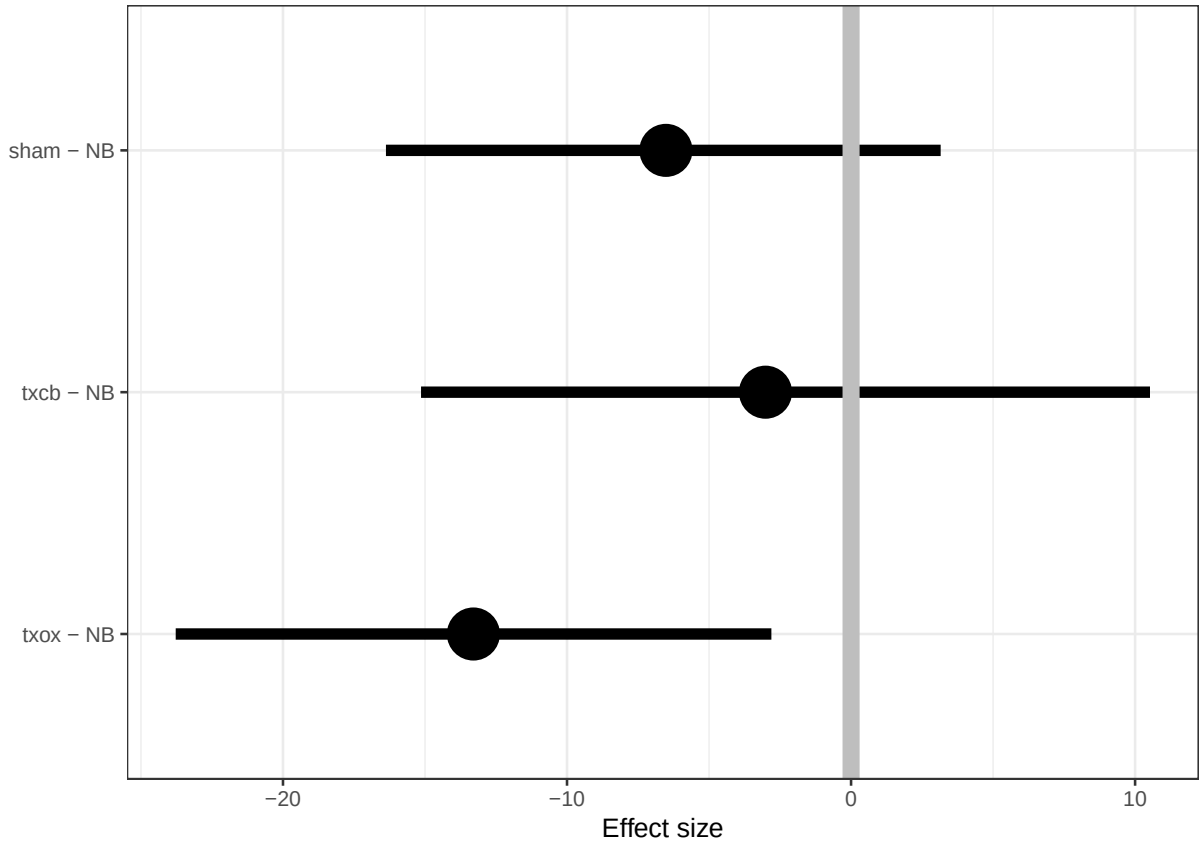
```
# Coefficient plot.
coeff_plot_data = interval_data(ba_seconds_fit)
plot_intervals(coeff_plot_data)
```



The first thing that must be noticed is that the grouptxox estimate is negative, its 95%-CI is all smaller than 0. This indicates that, on average, the treatment credibly reduces the duration of BA episodes by some amount.

To estimate by what amount (effect size) we perform contrasts. To do this, we simply postprocess the completed MCMC chain and obtain the posterior distribution of the differences between the NB and each of the other groups, respectively. This is performed using the function credible_intervals_data described in the apendix.

```
# Contrasts.
credible_intervals_dat = credible_intervals_data(ba_seconds,ba_seconds_fit)

# Plot contrasts.
plot_contrasts(credible_intervals_dat,xlab="Effect size")
```



The discussion of these findings is presented in the manuscript.

Total number of BA cases model per hour.

Data preparation.

```
# Table available number of hours-data per mice.
mice_totalhours <- group_by(data,mouse) %>%
                   summarise(totaltime = sum(seconds)/3600)

# Join data and mice_hours.
dataa <- right_join(data,mice_totalhours,by="mouse")

# Final data for model fit.
ba_counts <- dataa %>%
             mutate(mouse=paste0(group," ",mouse)) %>%
             group_by(mouse) %>%
             summarise(BA_counts = sum(behavior == "BA"),hours=min(totaltime)) %>%
             separate(mouse,c("group","mouse"))

# Change variable types.
ba_counts$group <- as.factor(ba_counts$group)
ba_counts$mouse <- as.factor(ba_counts$mouse)
```

Model fit.

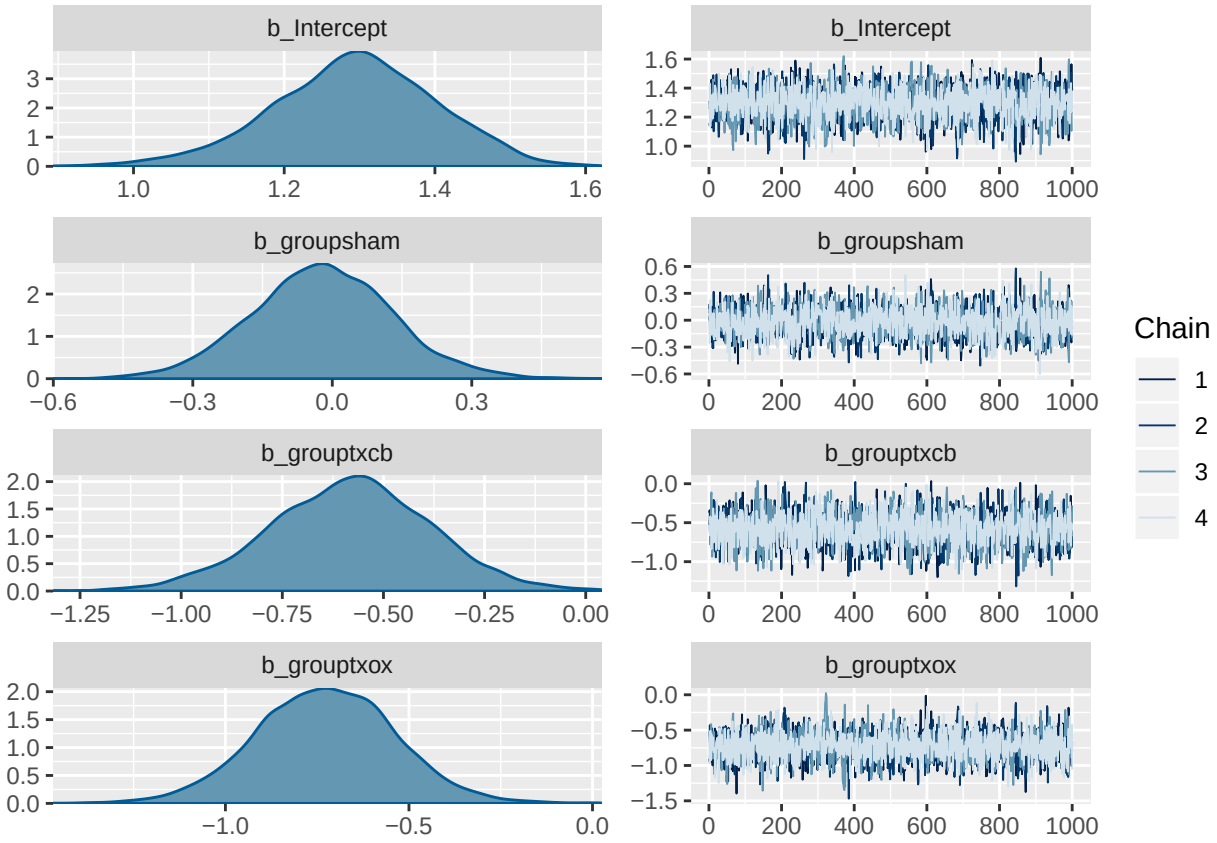
```
# Fit brms model for number of BA attacks per hour.
ba_counts_fit <- brm(BA_counts ~ group + offset(log(hours)),
                     data = ba_counts, family = poisson(),
                     control = list(adapt_delta = 0.9999,max_treedepth=15))
```

No runtime warnings during estimation were thrown which is a first indication of proper convergence.

We now perform further convergence checks.

First we see if the trace plot exhibits convergence.

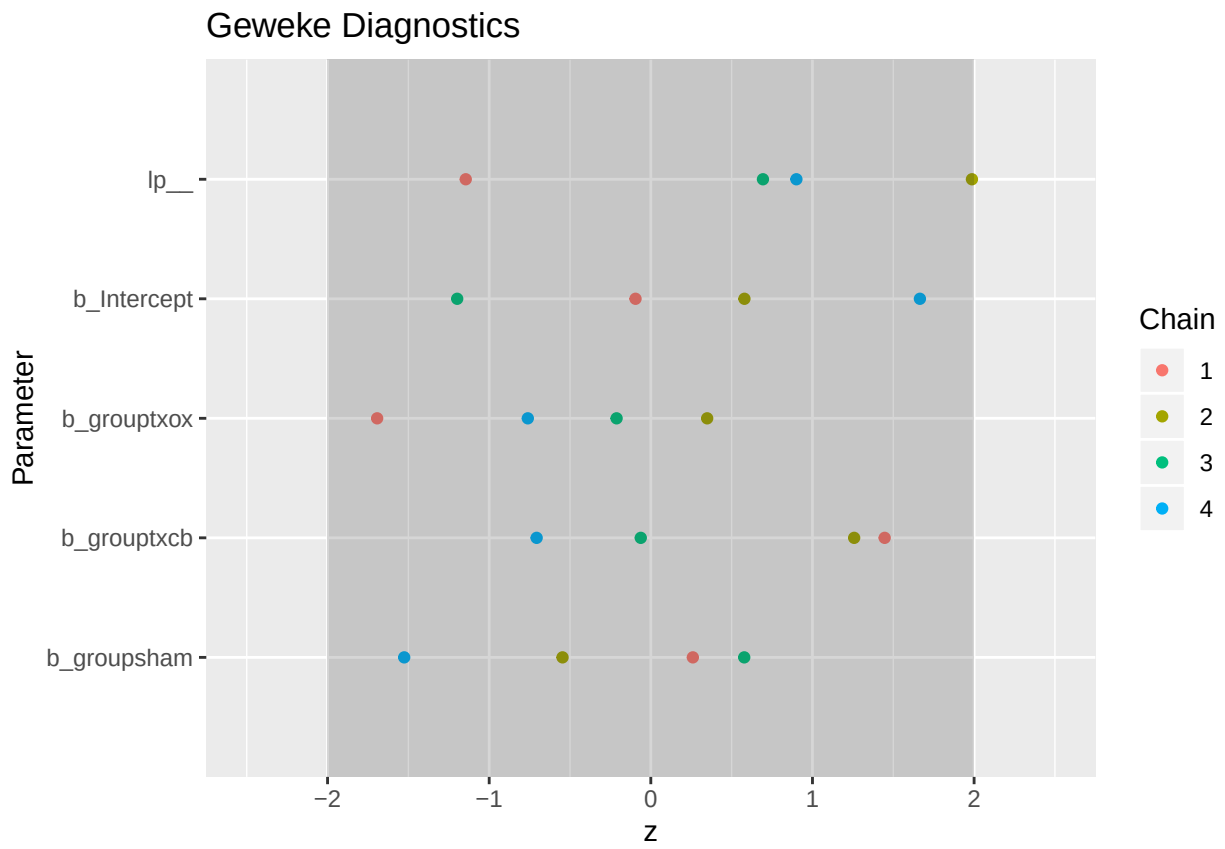
```
plot(ba_counts_fit)
```



Looks ok.

Geweke Diagnostic.

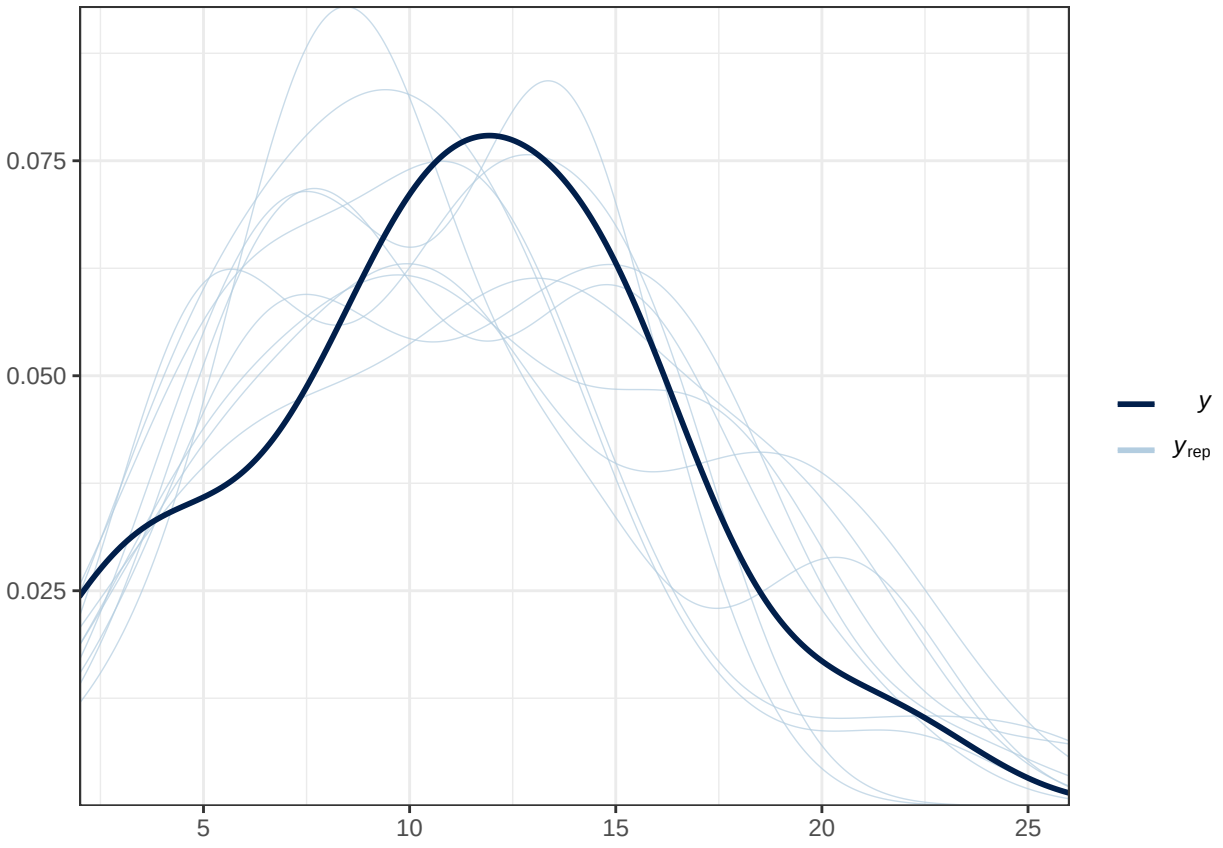
```
mcmc_df = ggs(as.mcmc(ba_counts_fit))

ggs_geweke(mcmc_df,frac1 = 0.1, frac2 = 0.5)
```



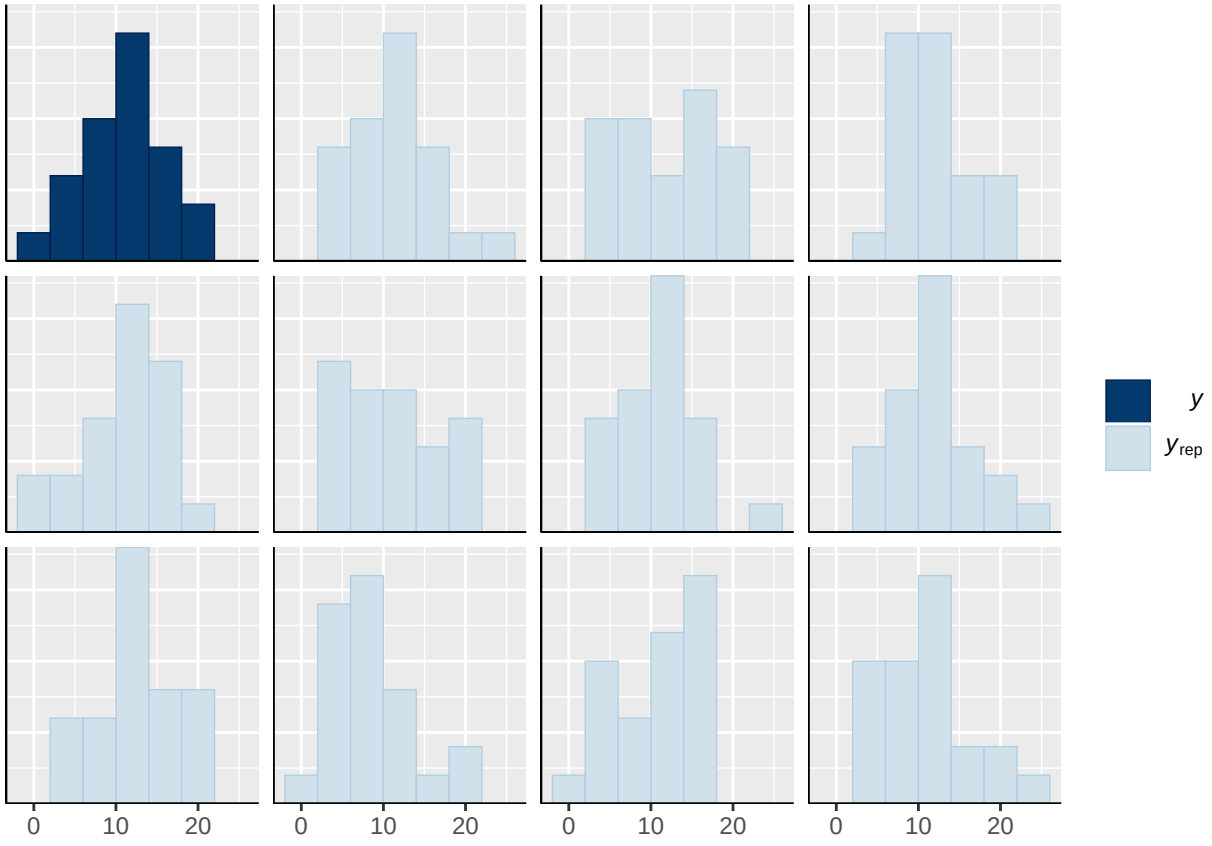
Ok.

We now perform Posterior Predictive checks.

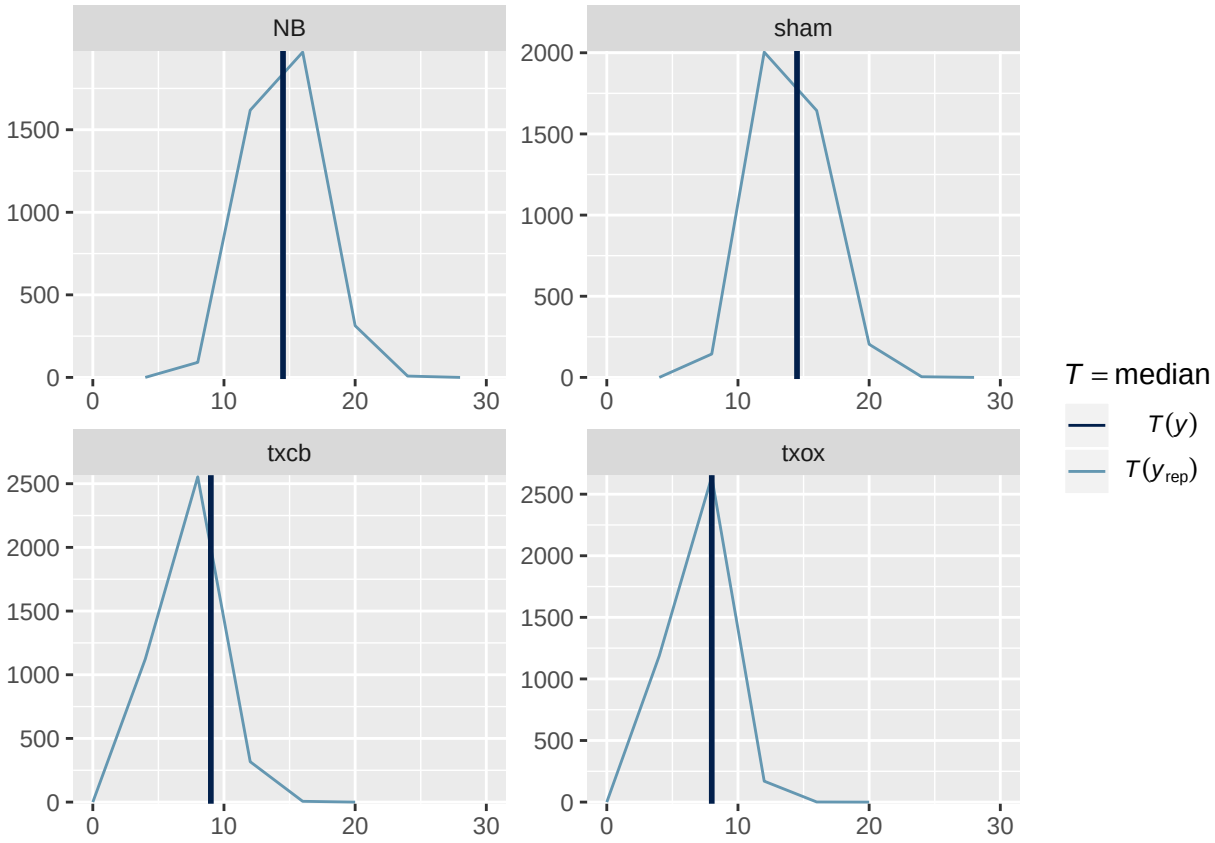
```
# Posterior Predictive checks.

# PP kernel density plot (observed vs simulated values).
pp <- brms::pp_check(ba_counts_fit,nsamples=10)
pp + theme_bw()
```




```
# PP histogram plot (observed vs simulated values).
y_simulated <- posterior_predict(ba_counts_fit, draws = 1000)
ppc_hist(ba_counts$BA_counts, y_simulated[1:11,], binwidth = 4)
```




```
# PP frecuency polygons vs the observed median of each group.
ppc_stat_freqpoly_grouped(ba_counts$BA_counts, y_simulated,
                          group=ba_counts$group, stat = "median", binwidth = 4)+
                          coord_cartesian(xlim = c(0, 30))
```



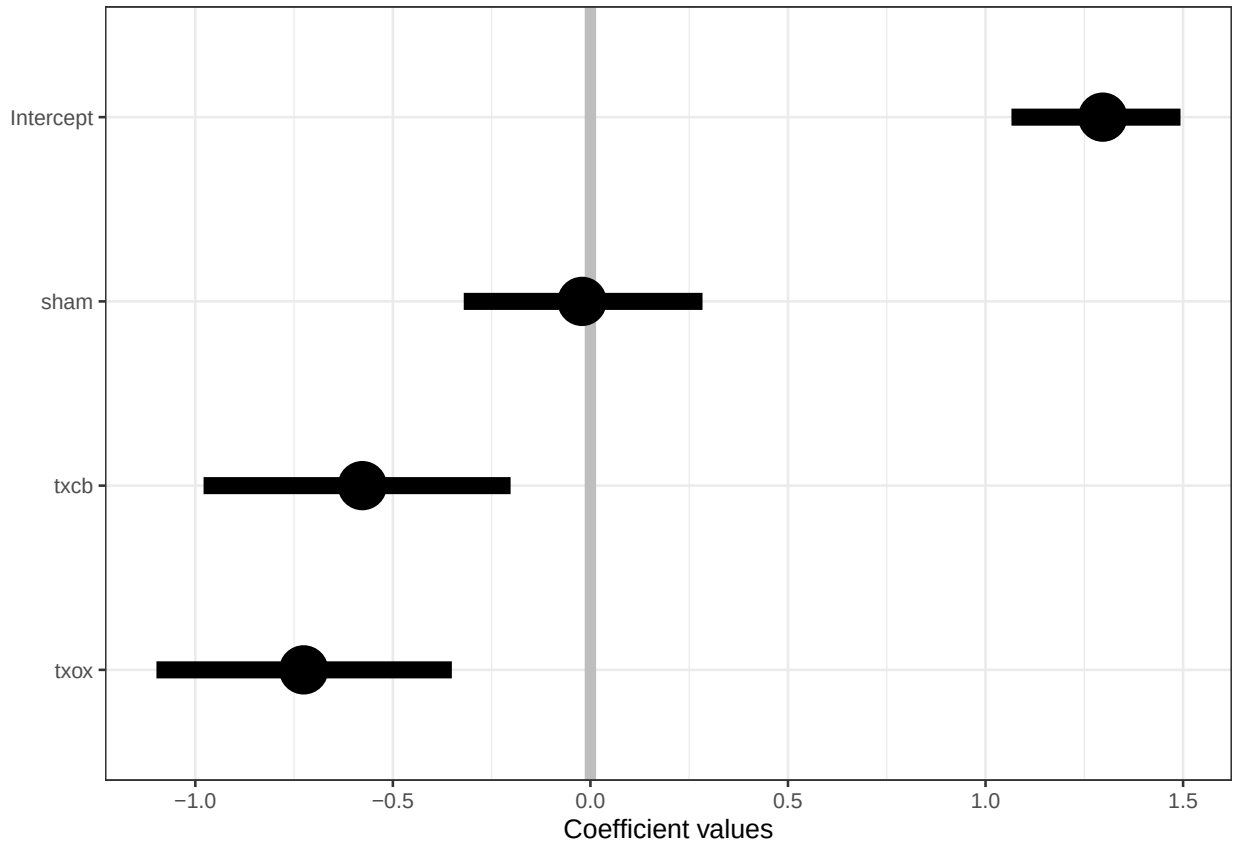
Densities and histograms of observed and predicted values are acceptably similar.

We proceed with model interpretation. First we produce the summary and coefficient plot for the model.

```
# Model summary.
summary(ba_counts_fit)
```

```
##  Family: poisson
##   Links: mu = log
## Formula: BA_counts ~ group + offset(log(hours))
##    Data: ba_counts (Number of observations: 23)
## Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup samples = 4000
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
## Intercept     1.29      0.11     1.07     1.49       2130 1.00
## groupsham    -0.02      0.15    -0.32     0.28       2193 1.00
## grouptxcb    -0.58      0.20    -0.98    -0.20       2278 1.00
## grouptxox    -0.73      0.19    -1.10    -0.35       1979 1.00
##
## Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
## is a crude measure of effective sample size, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

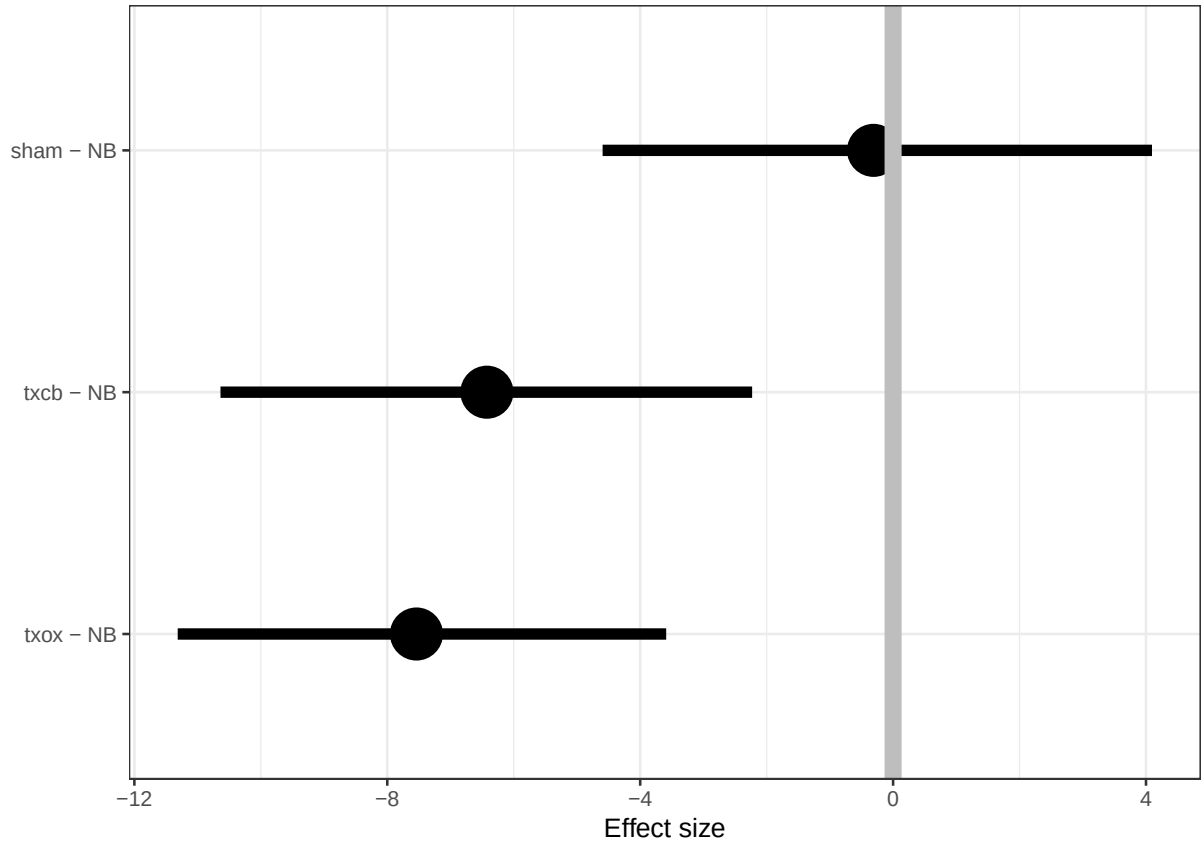
```
# Coefficient plot.
coeff_plot_data = interval_data(ba_counts_fit)
plot_intervals(coeff_plot_data)
```



It can be see that both the grouptxox and the grouptxcb estimates are negative (95%-CI's are smaller than 0). Indicating that those treatments credibly reduce the BA count per hour by some amount.

We perform contrasts to measure their effect sizes.

```
# Contrasts.
credible_intervals_dat = credible_intervals_data_hours(ba_counts,ba_counts_fit)

# Plot contrasts.
plot_contrasts(credible_intervals_dat,xlab="Effect size")
```



Percentage of total time spent in a BA state.

Data preparation.

```
# Total hours of sampling by mouse.
mice_totalseconds <- group_by(data,mouse) %>%
  summarise(totaltime = sum(seconds))

# Join data and mice_hours.
dataa <- right_join(data,mice_totalseconds,by="mouse")

# Final data for model fit.
ba_percentage <- filter(dataa,behavior=="BA") %>%
  mutate(mouse=paste0(group," ",mouse)) %>%
  group_by(mouse) %>%
  summarise(BA_totaltime = sum(seconds),hours=min(totaltime)) %>%
  mutate(percentage_time_BA=(BA_totaltime/hours)) %>%
  separate(mouse,c("group","mouse"))

# Change variable types.
ba_percentage$group <- as.factor(ba_percentage$group)
ba_percentage$mouse <- as.factor(ba_percentage$mouse)
```

Model fit.

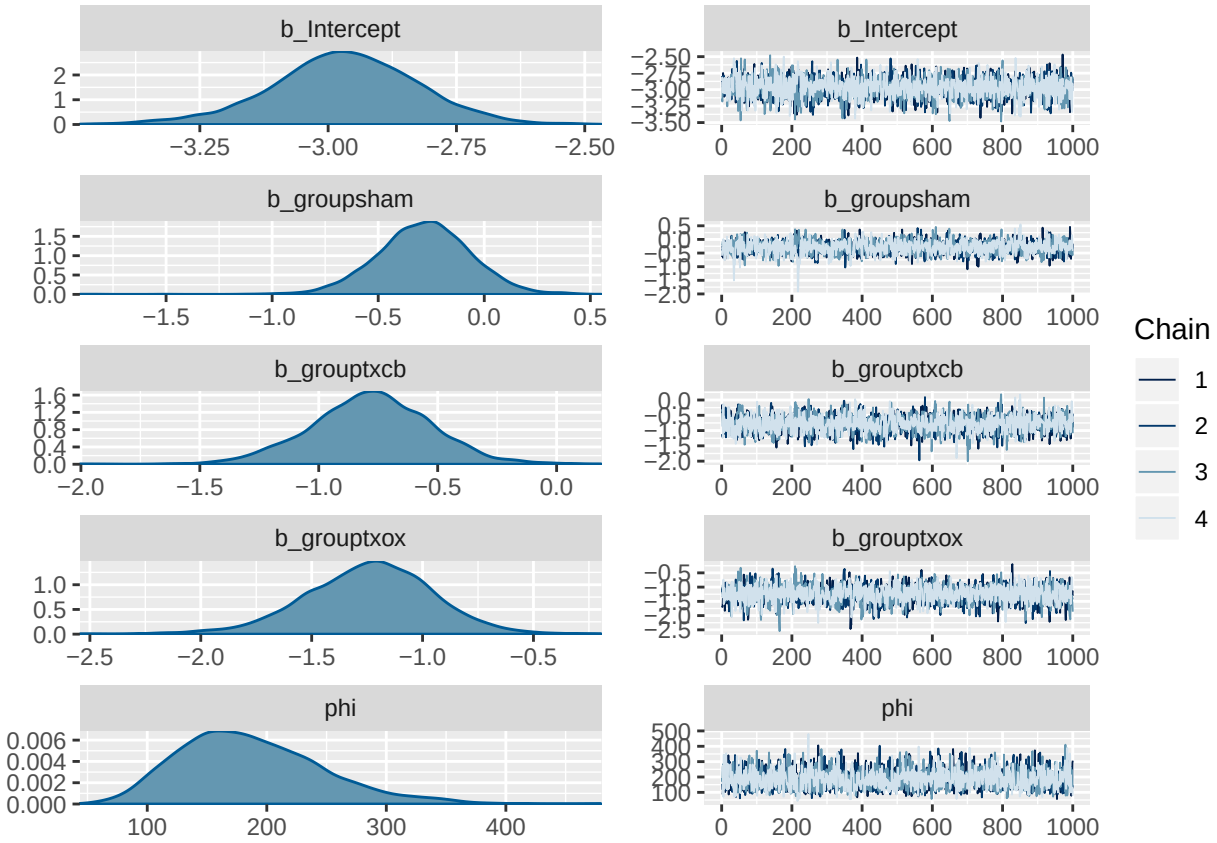
```
# Fit brms model for total time (%) spent in BA state.
ba_totaltime_fit <- brm(percentage_time_BA ~ group ,
                        data = ba_percentage,
                        family = Beta,
                        control = list(adapt_delta = 0.999,max_treedepth = 15))
```

No runtime warnings during estimation were thrown which is a first indication of proper convergence.

We now perform further convergence checks.

First we see if the trace plot exhibits convergence.

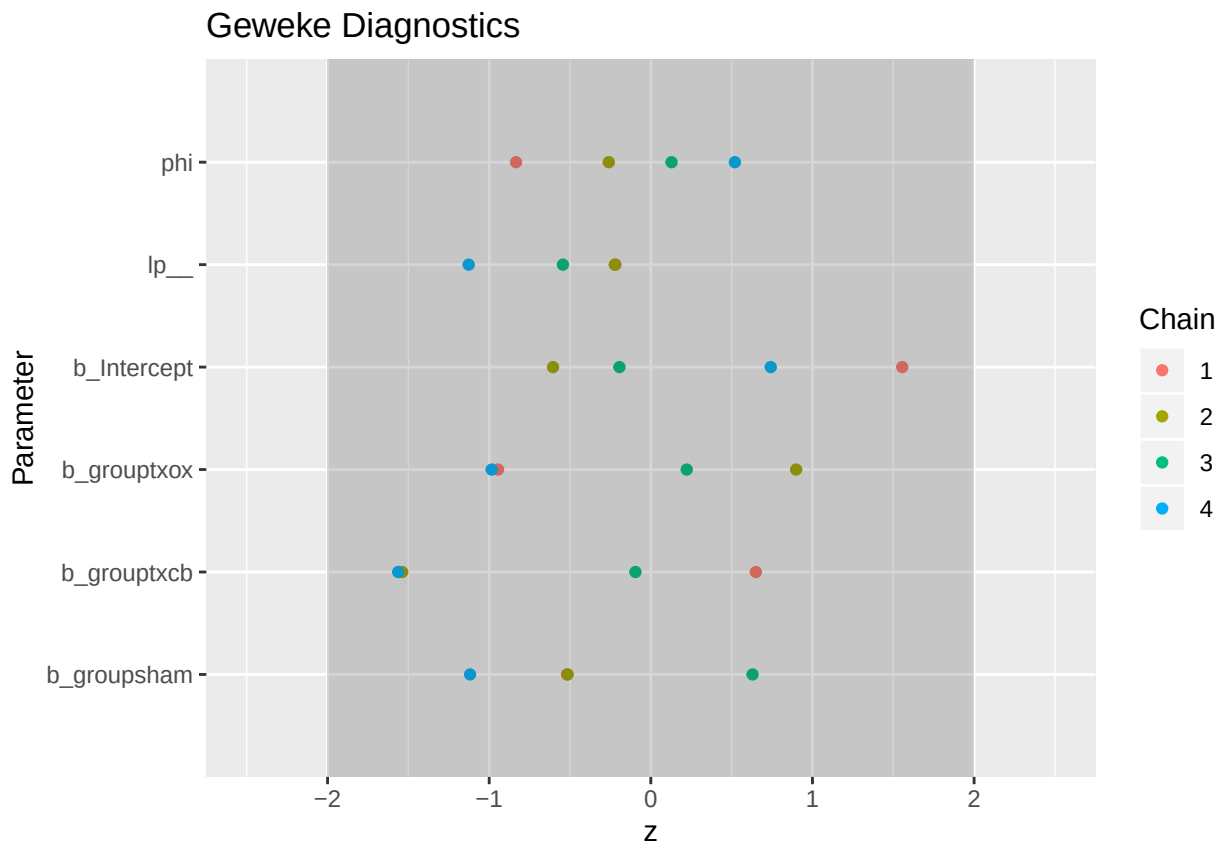
```
plot(ba_totaltime_fit)
```



Looks ok.

Geweke Diagnostic.

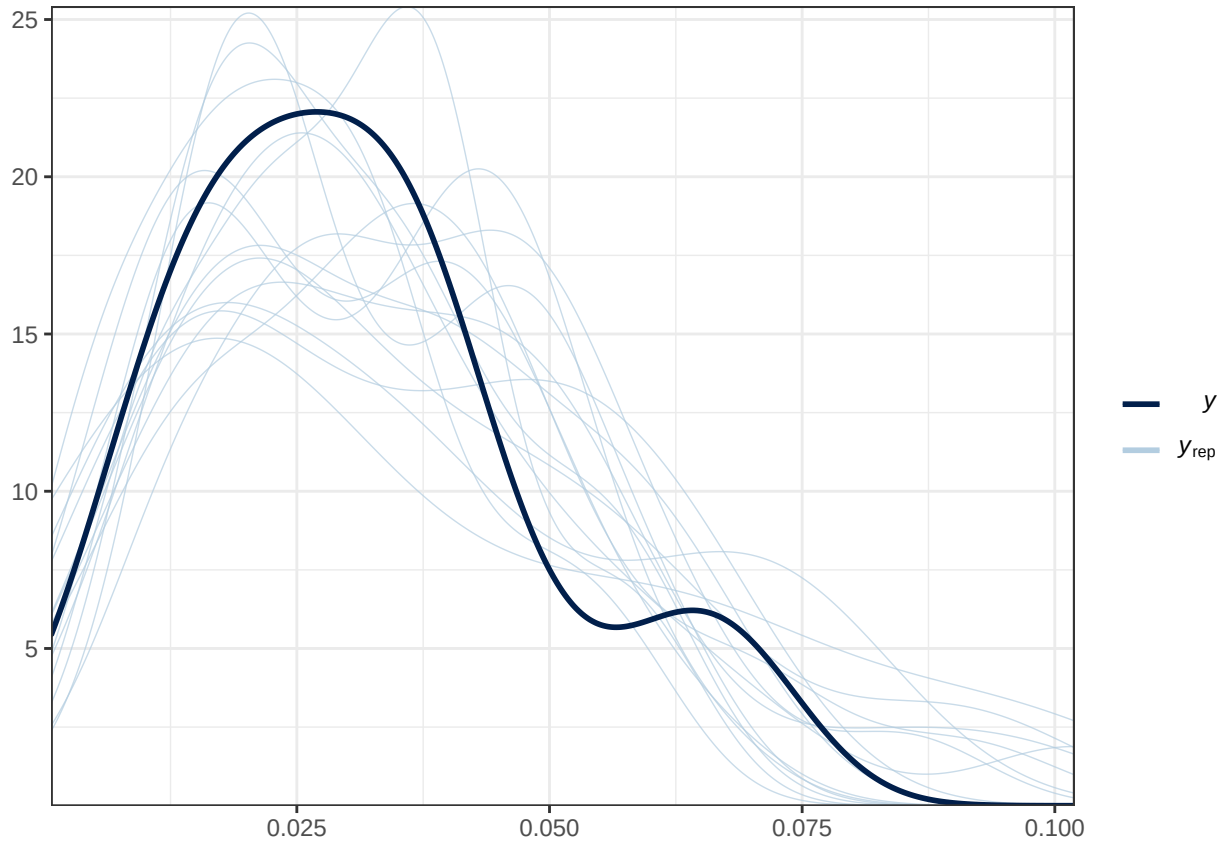
```
mcmc_df = ggs(as.mcmc(ba_totaltime_fit))

ggs_geweke(mcmc_df,frac1 = 0.1, frac2 = 0.5)
```



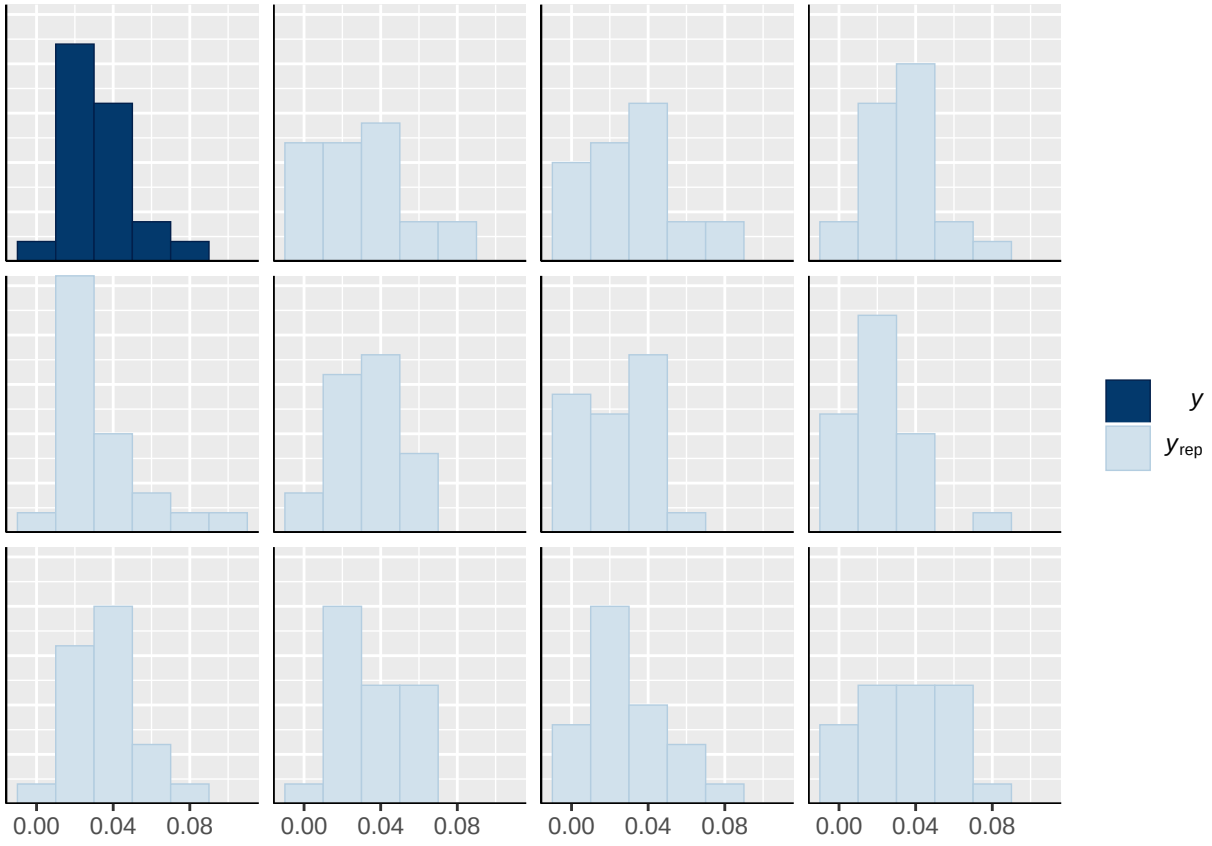
Ok.

We now perform Posterior Predictive checks.

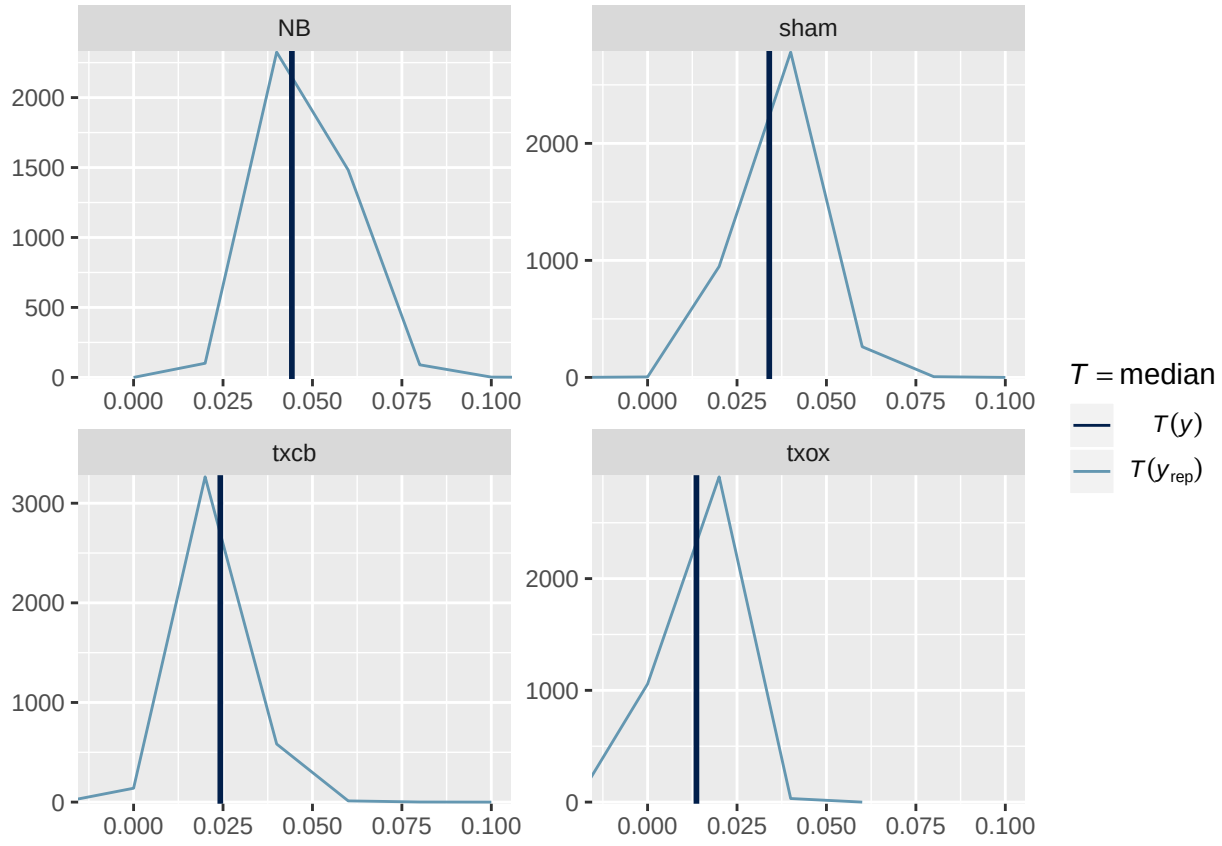
```
# Posterior Predictive checks.

# PP kernel density plot (observed vs simulated values).
pp <- brms::pp_check(ba_totaltime_fit,nsamples=15)
pp + theme_bw()
```




```
# PP histogram plot (observed vs simulated values).
y_simulated <- posterior_predict(ba_totaltime_fit, draws = 1000)
ppc_hist(ba_percentage$percentage_time_BA, y_simulated[1:11,], binwidth = 0.02)
```




```
# PP frecuency polygons vs the observed median of each group.
ppc_stat_freqpoly_grouped(ba_percentage$percentage_time_BA, y_simulated,
                          group=ba_percentage$group, stat = "median", binwidth = 0.02)+
                          coord_cartesian(xlim = c(-0.01, 0.1))
```



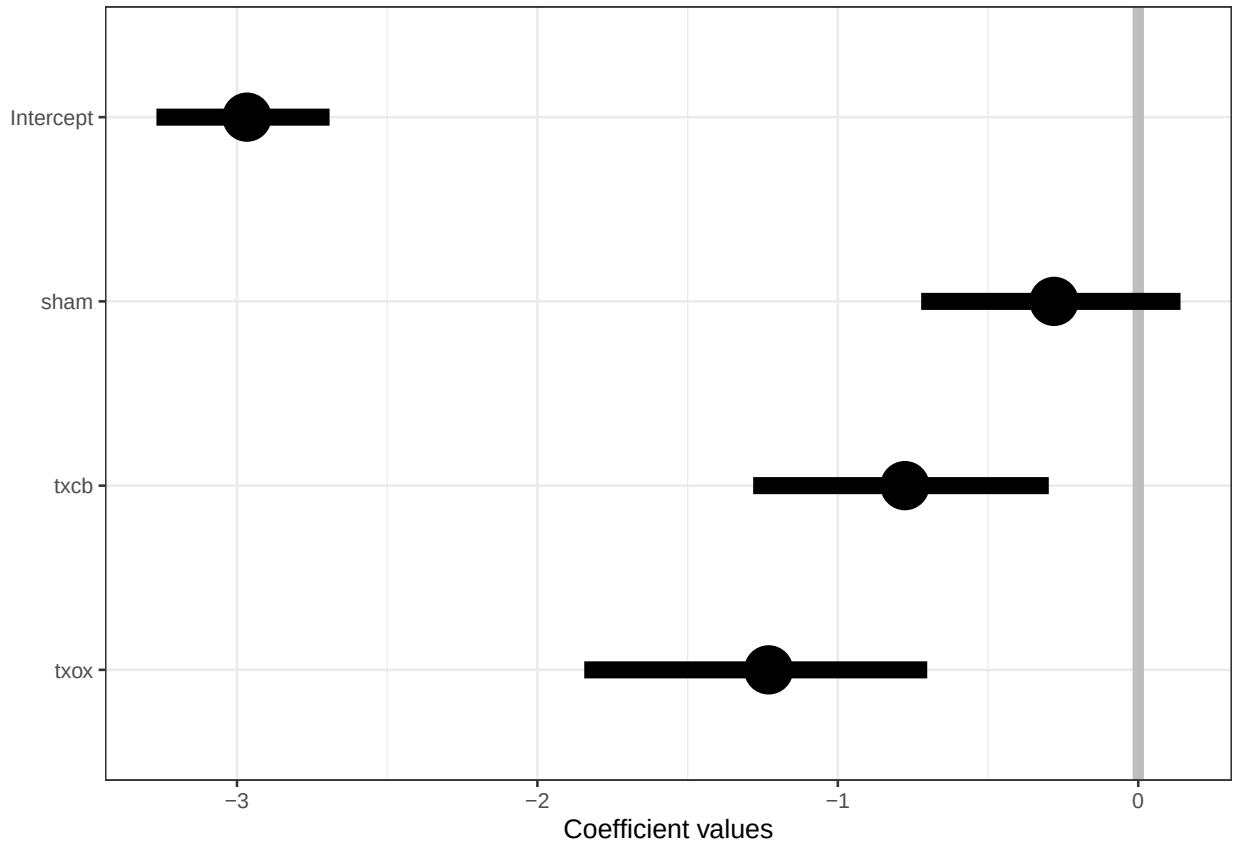
Densities and histograms of observed and predicted values are acceptably similar.

We proceed with model interpretation. First we produce the summary and coefficient plot for the model.

```
# Model summary.
summary(ba_counts_fit)
```

```
##  Family: poisson
##   Links: mu = log
## Formula: BA_counts ~ group + offset(log(hours))
##    Data: ba_counts (Number of observations: 23)
## Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup samples = 4000
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
## Intercept     1.29      0.11     1.07     1.49       2130 1.00
## groupsham    -0.02      0.15    -0.32     0.28       2193 1.00
## grouptxcb    -0.58      0.20    -0.98    -0.20       2278 1.00
## grouptxox    -0.73      0.19    -1.10    -0.35       1979 1.00
##
## Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
## is a crude measure of effective sample size, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

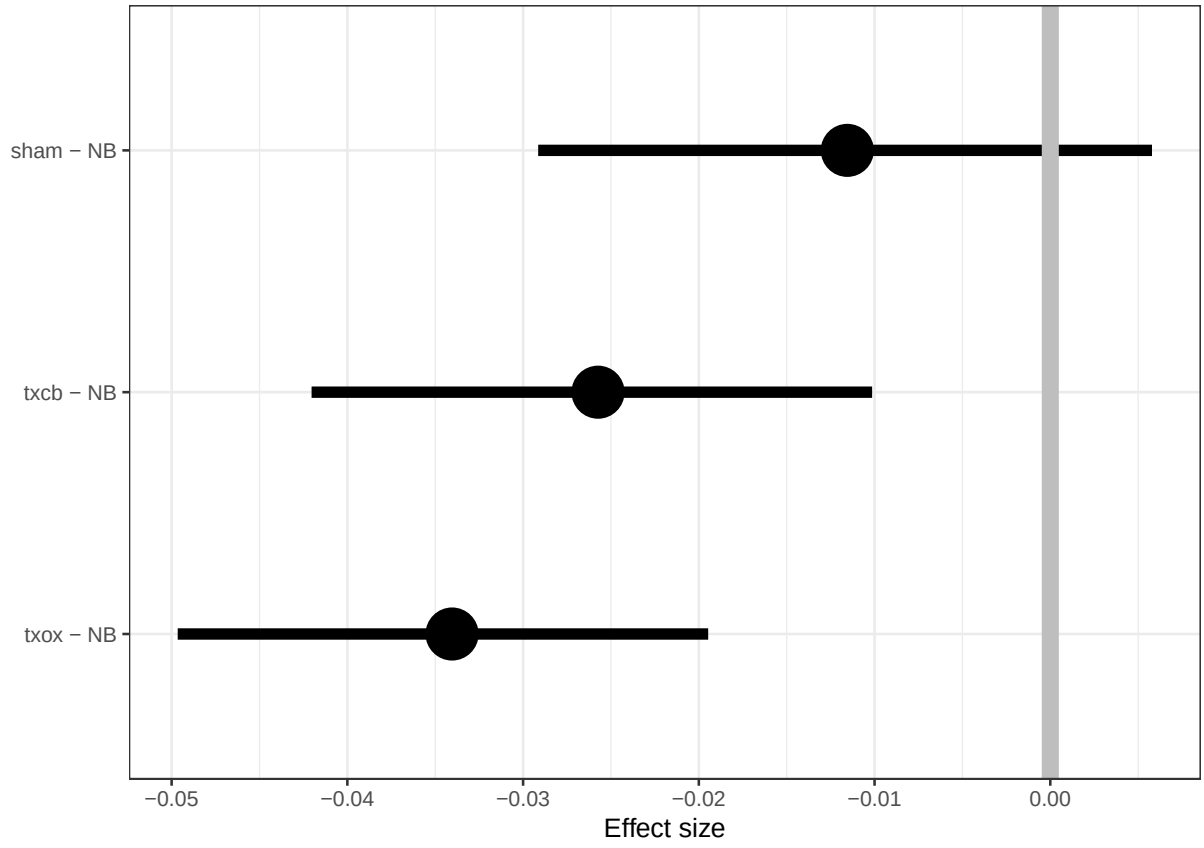
```
# Coefficient plot.
coeff_plot_data = interval_data(ba_totaltime_fit)
plot_intervals(coeff_plot_data)
```



It can be see that both the grouptxox and the grouptxcb estimates are negative (95%-CI's are smaller than 0). Indicating that those treatments credibly reduce the BA count per hour by some amount.

We perform contrasts to measure their effect sizes.

```
# Contrasts.
credible_intervals_dat = credible_intervals_data_hours(ba_percentage,ba_totaltime_fit)

# Plot contrasts.
plot_contrasts(credible_intervals_dat,xlab="Effect size")
```



Apendix (custom functions)

```
# Load packages.
library("ggplot2")
library("multcomp")
library("broom")

#################################################################################################
# Function to create interval data for plotting.
interval_data <- function(brms_model_fit,columns=1:4,
                                        param_names=c("Intercept","sham","txcb","txox"),
                                        prob=0.5,
                                        prob_outer=0.95,
                                        point_est="median"){

  post <- posterior_samples(brms_model_fit)[,columns]
  colnames(post) <- param_names
  interval_df <- mcmc_intervals_data(x=post,prob=prob,
                                              prob_outer=prob_outer,
                                              point_est=point_est)
  return(interval_df)
}
```

```
#################################################################################################
# Function to produce a coefficient plot (credible intervals).
plot_intervals <- function(data, draw_points = TRUE,
                           draw_ref_line = TRUE,
                           line_position = 0){

  maybe_points <- if (draw_points) data else data[0, ]
  geom_maybe_vline <- if (draw_ref_line) geom_vline else geom_ignore

  ggplot(data) +
    aes_(y = ~ parameter, yend = ~ parameter) +
    geom_maybe_vline(xintercept = line_position, size = 2, color = "gray") +
    geom_segment(aes_(x = ~ ll, xend = ~ hh), size = 3) +
    geom_point(aes_(x = ~ m), data = maybe_points, size = 8) +
    scale_y_discrete(limits = rev(data$parameter)) +
    labs(x = "Coefficient values", y = NULL)+
    theme_bw(base_size = 40)
}

#################################################################################################
# Function to create data for contrasts between treatments.
contrasts_data <- function(model_input_data
                           ,brms_model_fit){

      posterior_data <- as.data.frame(fitted(brms_model_fit,
                                             data.frame(group = levels(model_input_data$group)),
                                             re_formula = NA,
                                             summary = FALSE))
      colnames(posterior_data) <- unique(model_input_data$group)
      return(posterior_data)
}

#################################################################################################
# Function to create data for contrasts between treatments
# (hours variable, needs 1+ because of offset).
contrasts_data_hours <- function(model_input_data,
                                 brms_model_fit){

      posterior_data <- as.data.frame(fitted(brms_model_fit,
                                             data.frame(group = levels(model_input_data$group)),
                                             hours=rep(4,4), # assumes 4 hours of sampling
                                             re_formula = NA,
                                             summary = FALSE))
      colnames(posterior_data) <- unique(model_input_data$group)
      return(posterior_data)
}

#################################################################################################
# Create data for interval plots.
credible_intervals_data <- function(data,fit,
                                    untreated="NB",
                                    columns=1:3){
```

```
  contrasts_dat <- contrasts_data(data[,columns],fit)
  treatments <- as.character(unique(data$group))
  treatments <- treatments[treatments!=untreated]
  output <- data.frame(matrix(0,length(treatments),8))
  names(output)=c("term", "estimate", "conf.low", "conf.high","median",
                  "ctrl_estimate","treatment_estimate","reduction_percentage")

  for (i in 1:nrow(output)){

    treatment <- treatments[i]
    output[i,"term"] <- paste0(treatment," - ",untreated)

    NB_vs_treatment <- contrasts_dat[,treatment]-contrasts_dat[,untreated]

    NB <- contrasts_dat[,untreated]
    treat <-  contrasts_dat[,treatment]

    quants <- quantile(NB_vs_treatment, probs = c(.5, .025, .975))

    output[i,"estimate"] <- quants[1]
    output[i,"conf.low"] <- quants[2]
    output[i,"conf.high"] <- quants[3]
    output[i,"median"] <- median(NB_vs_treatment)
    output[i,"ctrl_estimate"] <- median(NB)
    output[i,"treatment_estimate"] <- median(treat)
    output[i,"reduction_percentage"] <- output[i,"median"]/output[i,"ctrl_estimate"]

  }

  output$term <- factor(output$term,levels=rev(output$term))

  return(output)
}

#############################################################################################
# Create data for interval plots
# (hours variable, needs 1+ because of offset).
credible_intervals_data_hours <- function(data,fit,
                                          untreated="NB",
                                          columns=1:3){

  contrasts_dat <- contrasts_data_hours(data[,columns],fit)
  treatments <- as.character(unique(data$group))
  treatments <- treatments[treatments!=untreated]
  output <- data.frame(matrix(0,length(treatments),8))
  names(output) <- c("term", "estimate", "conf.low", "conf.high","median",
                     "ctrl_estimate","treatment_estimate","reduction_percentage")

  for (i in 1:nrow(output)){

    treatment <- treatments[i]
    output[i,"term"] <- paste0(treatment," - ",untreated)
```

```
    NB_vs_treatment <- contrasts_dat[,treatment]-contrasts_dat[,untreated]

    NB <- contrasts_dat[,untreated]
    treat <- contrasts_dat[,treatment]

    quants <- quantile(NB_vs_treatment, probs = c(.5, .025, .975))

    output[i,"estimate"] <- quants[1]
    output[i,"conf.low"] <- quants[2]
    output[i,"conf.high"] <- quants[3]
    output[i,"median"] <- median(NB_vs_treatment)
    output[i,"ctrl_estimate"] <- median(NB)
    output[i,"treatment_estimate"] <- median(treat)
    output[i,"reduction_percentage"] <- output[i,"median"]/output[i,"ctrl_estimate"]

  }

  output$term <- factor(output$term,levels=rev(output$term))

  return(output)
}

##############################################################################################
# Plot pairwise comparisons (contrasts).
plot_contrasts <- function(comps,
                           xlab="Effect size"){

  ggplot(comps, aes(y = estimate, x = term)) +
    geom_pointrange(aes(ymin = conf.low,ymax = conf.high),size=2) +
    geom_hline(yintercept = 0, size = 3, color = "gray") +
    scale_y_continuous(xlab) + scale_x_discrete("") + coord_flip() +
    theme_bw(base_size = 40)
}
##############################################################################################
```

References

Bürkner, P.-C., 2017. brms: An R Package for Bayesian Multilevel Models Using Stan. J. Stat. Softw. 80. https://doi.org/10.18637/jss.v080.i01

Bürkner, P.-C., Vuorre, M., 2019. Ordinal Regression Models in Psychology: A Tutorial. Adv. Methods Pract. Psychol. Sci. 2, 77-101. https://doi.org/10.1177/2515245918823199

Carpenter, B., Gelman, A., Hoffman, M.D., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M., Guo, J., Li, P., Riddell, A., 2017. Stan: A Probabilistic Programming Language. J. Stat. Softw. 76. https://doi.org/10.18637/jss.v076.i01

Gabry, J. , Simpson, D. , Vehtari, A. , Betancourt, M. and Gelman, A., 2019. Visualization in Bayesian workflow. J. R. Stat. Soc. A, 182: 389-402. doi:10.1111/rssa.12378

Gelman, A., 2006. Prior distributions for variance parameters in hierarchical models. Bayesian Anal. 1, 515-533.

Geweke, J., 1992. Evaluating the Accuracy of Sampling-Based Approaches to Calculating Posterior Moments. In Bayesian Statistics 4 (ed JM Bernardo, JO Berger, AP Dawid, and AFM Smith). Clarendon Press, Oxford,

UK.

Lesaffre, E. and Lawson, A. B., 2012. Bayesian Biostatistics. New York: John Wiley & Sons, Ltd. 534 pages, ISBN 978-0-470-01823-1.