Piecewise Regression using STAN Directly

Michael Gilchrist

date: 2022-11-29

Goal

- Fit one and two piece Poisson and Quasiposson GLM

Recap

Set up

Install libraries

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

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


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

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

Load Data

```
load(file.path("input", "data.processing_2022-11-09.Rda"),
     verbose = TRUE)
```

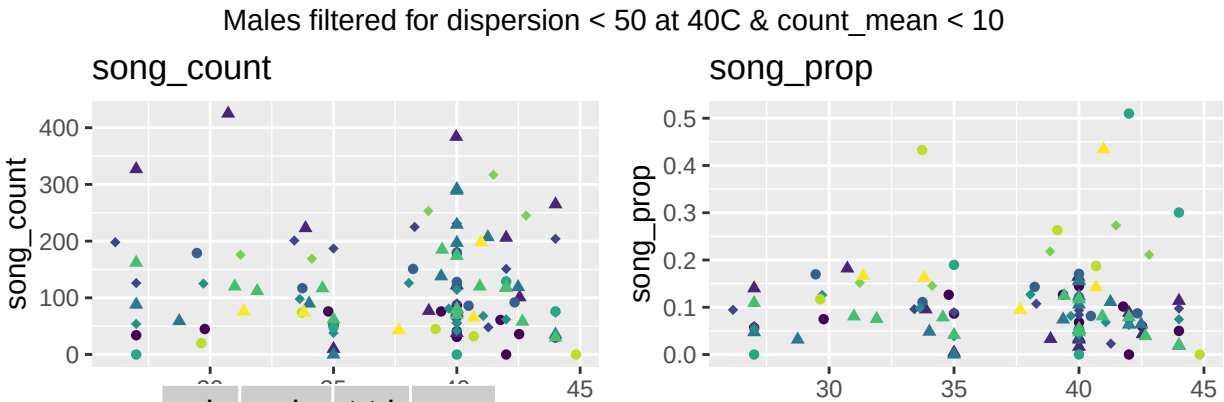
```
## Loading objects:
##   song_data
##   song_data_40C
##   song_stats
##   song_stats_40C
##   bird_bill_data
```

Examine Data

Create Working Dataset

```
filter_data <- TRUE

if(filter_data) {
  males_filtered_disp <- song_stats_40C %>%
    filter(dispersion < 50) %>%
    pull(male)

  males_filtered_mean <- song_stats %>%
    filter(mean > 10) %>%
    pull(male)

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


##males_selected <-

data_ind <- song_data %>%
  filter(male %in% male_vector) %>%
  arrange(male) %>%
  ##    left_join(male_shape, by = "male") %>%
  mutate()

stats_ind <- song_stats %>%
  filter(male %in% male_vector)
```

Plot song_count



	male	n_obs	total	mean
1	T236	13	2095	161
2	T243	13	1861	143
3	T246	5	253	51
4	T247	14	1487	106
5	T234	13	601	46
6	T244	13	993	76
7	T235	13	2333	179
8	T237	8	1055	132
9	T258	5	171	34
10	T257	5	1160	232

Analyze Data:

Simple GLM

```
glm_poisson_0 <- glm(song_count ~
                       1,
                     data = data_ind,
                     family = poisson(link = "identity")
                     )

summary(glm_poisson_0)
```

```
##
## Call:
## glm(formula = song_count ~ 1, family = poisson(link = "identity"),
##     data = data_ind)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -15.264   -6.300   -2.558    5.042   21.979
```

```
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  116.495      1.043   111.6   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 6451.2  on 106  degrees of freedom
## Residual deviance: 6451.2  on 106  degrees of freedom
## AIC: 7107.5
##
## Number of Fisher Scoring iterations: 3
```

```
glm_qpoisson_0 <- glm(song_count ~
                        1,
                      data = data_ind,
                      family = quasipoisson(link = "identity")
                      )

summary(glm_qpoisson_0)
```

```
##
## Call:
## glm(formula = song_count ~ 1, family = quasipoisson(link = "identity"),
##     data = data_ind)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -15.264   -6.300   -2.558    5.042   21.979
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  116.495      8.251   14.12   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasipoisson family taken to be 62.52277)
##
##     Null deviance: 6451.2  on 106  degrees of freedom
## Residual deviance: 6451.2  on 106  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 3
```

```
# Following analysis not needed
#
# glm_qpoisson_1 <- glm(song_count ~
#                         (1 + male),
#                       data = data_ind,
#                       family = quasipoisson(link = "identity")
#                       )
```

```
#
#
#summary(glm_qpoisson_1)
```

Stan

Set Up Data

```
song_count <- data_ind %>% pull(song_count)
temp <- data_ind %>% pull(temp)
N <- length(temp)
## parameters to be printed

pars <- c("t1", "y0")
pars_full <- c(pars, "b1", "lp__")
```

Normal error

```
iter <- 6000 ## increasing to 10k doesn't really help
fit_norm <- stan(file = "two.piece_normal.stan",
              model_name = "Two piece gaussian",
              data=list(x = temp, y = song_count,
                        N = N,
                        tmax = 46,
                        y_tmax = 0),
              chains = 4,
              cores = 4,
              iter = iter,
              warmup = iter/2)
```

```
## Warning: There were 8458 transitions after warmup that exceeded the maximum treedepth. Increase max_
## https://mc-stan.org/misc/warnings.html#maximum-treedepth-exceeded

## Warning: Examine the pairs() plot to diagnose sampling problems

## Warning: The largest R-hat is 2.38, indicating chains have not mixed.
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#r-hat

## Warning: Bulk Effective Samples Size (ESS) is too low, indicating posterior means and medians may be
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#bulk-ess

## Warning: Tail Effective Samples Size (ESS) is too low, indicating posterior variances and tail quant
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#tail-ess
```

- Even with a (poorly chosen) prior on sigma, this fails to converge. Here's some output.

```
> Chain 1: Iteration: 6000 / 6000 [100%]  (Sampling)
> Chain 1:
> Chain 1:  Elapsed Time: 15.038 seconds (Warm-up)
> Chain 1:                14.647 seconds (Sampling)
> Chain 1:                29.685 seconds (Total)
> Chain 1:
> Warning messages:
> 1: There were 1 divergent transitions after warmup. See
> https://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
> to find out why this is a problem and how to eliminate them.
> 2: There were 3819 transitions after warmup that exceeded the maximum treedepth. Increase max_treedep
> https://mc-stan.org/misc/warnings.html#maximum-treedepth-exceeded
> 3: Examine the pairs() plot to diagnose sampling problems
>
> 4: The largest R-hat is 1.55, indicating chains have not mixed.
> Running the chains for more iterations may help. See
> https://mc-stan.org/misc/warnings.html#r-hat
> 5: Bulk Effective Samples Size (ESS) is too low, indicating posterior means and medians may be unreli
> Running the chains for more iterations may help. See
> https://mc-stan.org/misc/warnings.html#bulk-ess
> 6: Tail Effective Samples Size (ESS) is too low, indicating posterior variances and tail quantiles ma
> Running the chains for more iterations may help. See
> https://mc-stan.org/misc/warnings.html#tail-ess
```

Poisson: Threshold (`t1`) and slope (`b`) formulation

- Model can be fitted
  - Data is overdispersed. (See comparison to `quasipoisson` below.)
  - Alternative formulation trying to estimate 'y0' directly does not currently work.

Null Model

```
iter <- 5000
tmax <- 46
t1min <- 25
t1max <- tmax - 0.001
t1min <- t1max - 0.001
## values to use for model predictions
tp = seq(25, t1max, length.out = 100)

fit_1 <- stan(file = "two.piece_poisson.stan",
              model_name = "one step poisson",
              data=list(t = temp,
                        y = song_count,
                        N = N,
                        tmax = tmax,
                        t1min = t1min,
                        t1max = t1max, ## max threshold value.
```

```
                        ## having it too close to tmax *sometimes* leads to sampling
                        ## near tmax, but with lower lp and very high E13) b1 values
                        y_tmax = 0,
                        tp = tp),
              chains = 4,
              cores = 4,
              iter = iter,
              warmup = floor(iter/2))

## Examine output
main <- "One piece"
print(main)
```

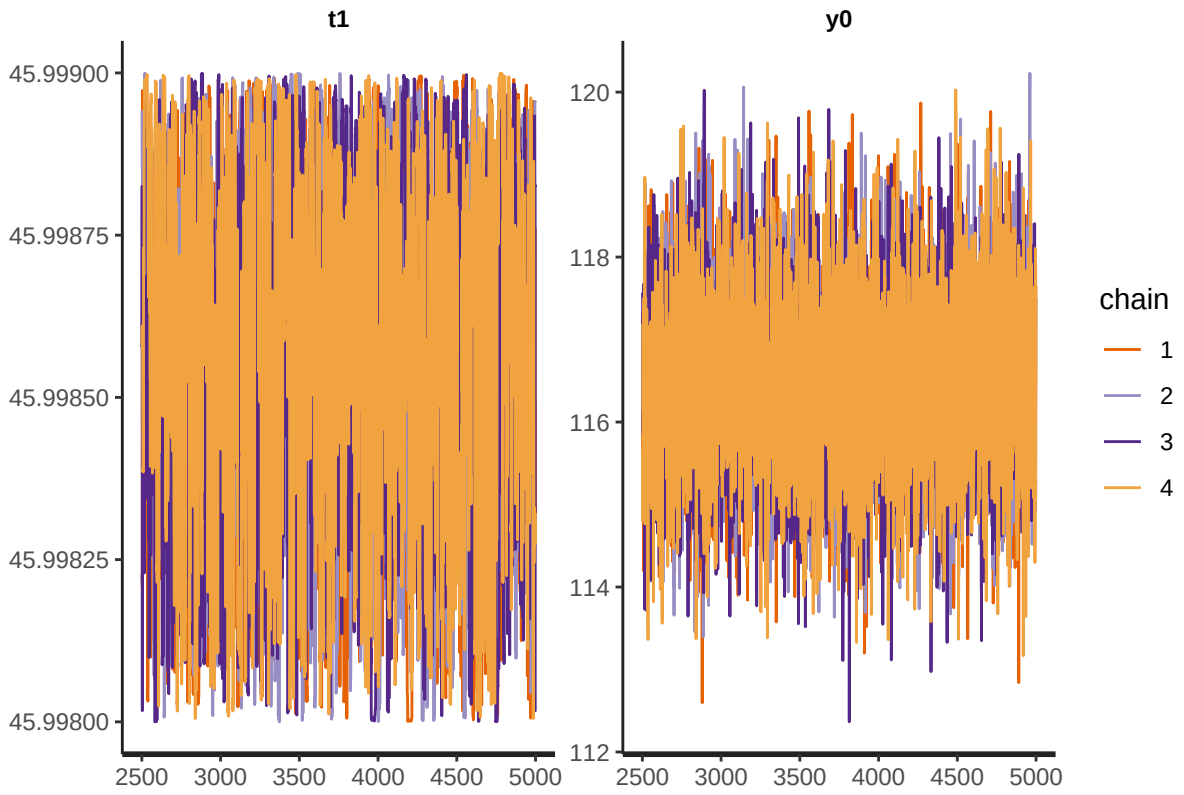
```
## [1] "One piece"
```

```
print(fit_1, pars = pars)
```

```
## Inference for Stan model: one step poisson.
## 4 chains, each with iter=5000; warmup=2500; thin=1;
## post-warmup draws per chain=2500, total post-warmup draws=10000.
##
##      mean se_mean   sd   2.5%    25%    50%    75%  97.5% n_eff Rhat
## t1  46.00    0.00 0.00  46.00  46.00  46.00  46.00  46.00  1316    1
## y0 116.48    0.01 1.04 114.47 115.78 116.48 117.18 118.51  8340    1
##
## Samples were drawn using NUTS(diag_e) at Fri Dec  2 14:49:30 2022.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
traceplot(fit_1, pars = pars, inc_warmup = FALSE, main = main)
```

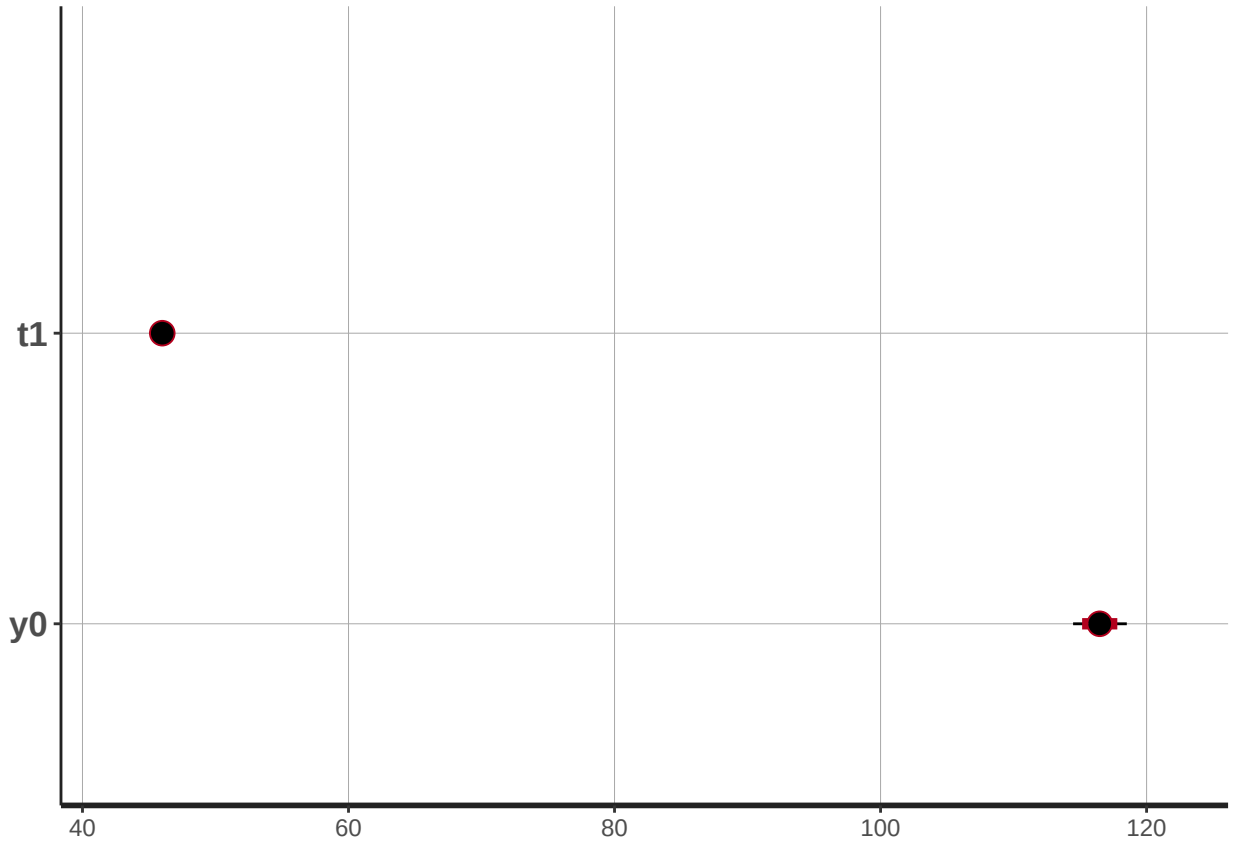
```
## Warning in ggplot2::geom_path(...): Ignoring unknown parameters: ‘main‘
```




```
plot(fit_1, pars = pars, main = main)
```

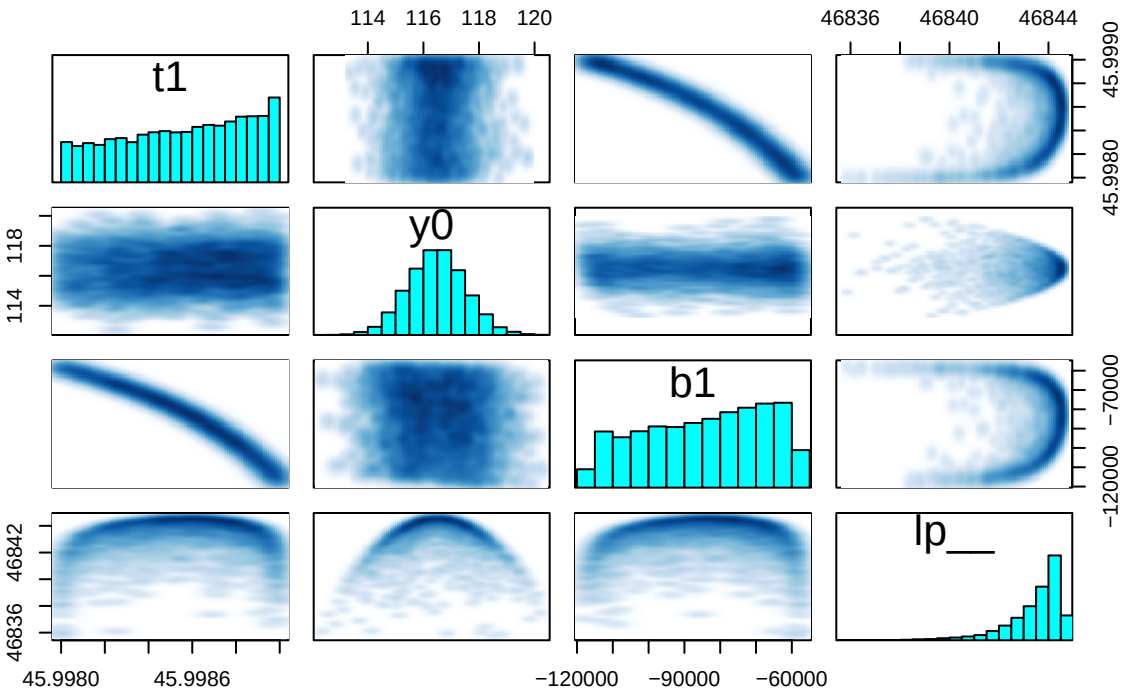
```
## ci_level: 0.8 (80% intervals)
```

```
## outer_level: 0.95 (95% intervals)
```




```
pairs(fit_1, pars = pars_full, main = main)
```

```
## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter
```




```
main = paste0("t1max = ", t1max)
```

Two Piece Model

- Examination of posterior pr values indicate this fits substantially better than the one piece model.
- LOO analysis below indicates it has more `problems`, however.

```
iter <- 5000
tmax <- 46
t1max <- tmax - 0.75;
t1min <- 25;
## values to use for model predictions
tp = seq(25, tmax, length.out = 100)

fit_2 <- stan(file = "two.piece_poisson.stan",
              model_name = "Two Piece poisson",
              data=list(t = temp,
                        y = song_count,
                        N = N,
                        tmax = tmax,
                        t1min = t1min,
                        t1max = t1max, ## max threshold value.
                        ## having it too close to tmax *sometimes* leads to sampling
                        ## near tmax, but with lower lp and very high E13) b1 values
                        y_tmax = 0,
                        tp = tp),
              chains = 4,
              cores = 4,
```

```
              iter = iter,
              warmup = floor(iter/2))

## Examine output
main <- "Two piece"
print(main)
```

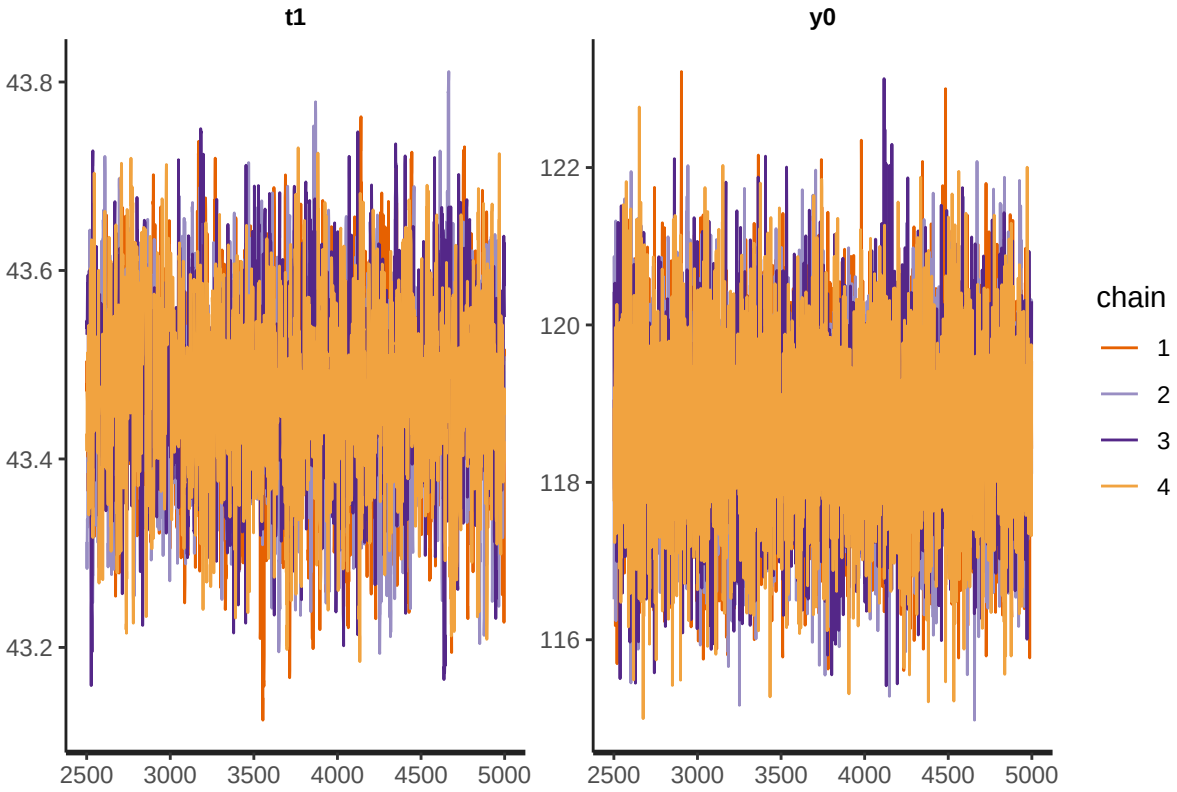
```
## [1] "Two piece"
```

```
print(fit_2, pars = pars)
```

```
## Inference for Stan model: one step poisson.
## 4 chains, each with iter=5000; warmup=2500; thin=1;
## post-warmup draws per chain=2500, total post-warmup draws=10000.
##
##      mean se_mean   sd   2.5%    25%    50%    75%  97.5% n_eff Rhat
## t1  43.47    0.00 0.10  43.27  43.41  43.48  43.54  43.66  1812    1
## y0 118.71    0.01 1.09 116.58 117.98 118.71 119.46 120.88  9511    1
##
## Samples were drawn using NUTS(diag_e) at Fri Dec  2 14:49:42 2022.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
traceplot(fit_2, pars = pars, inc_warmup = FALSE, main = main)
```

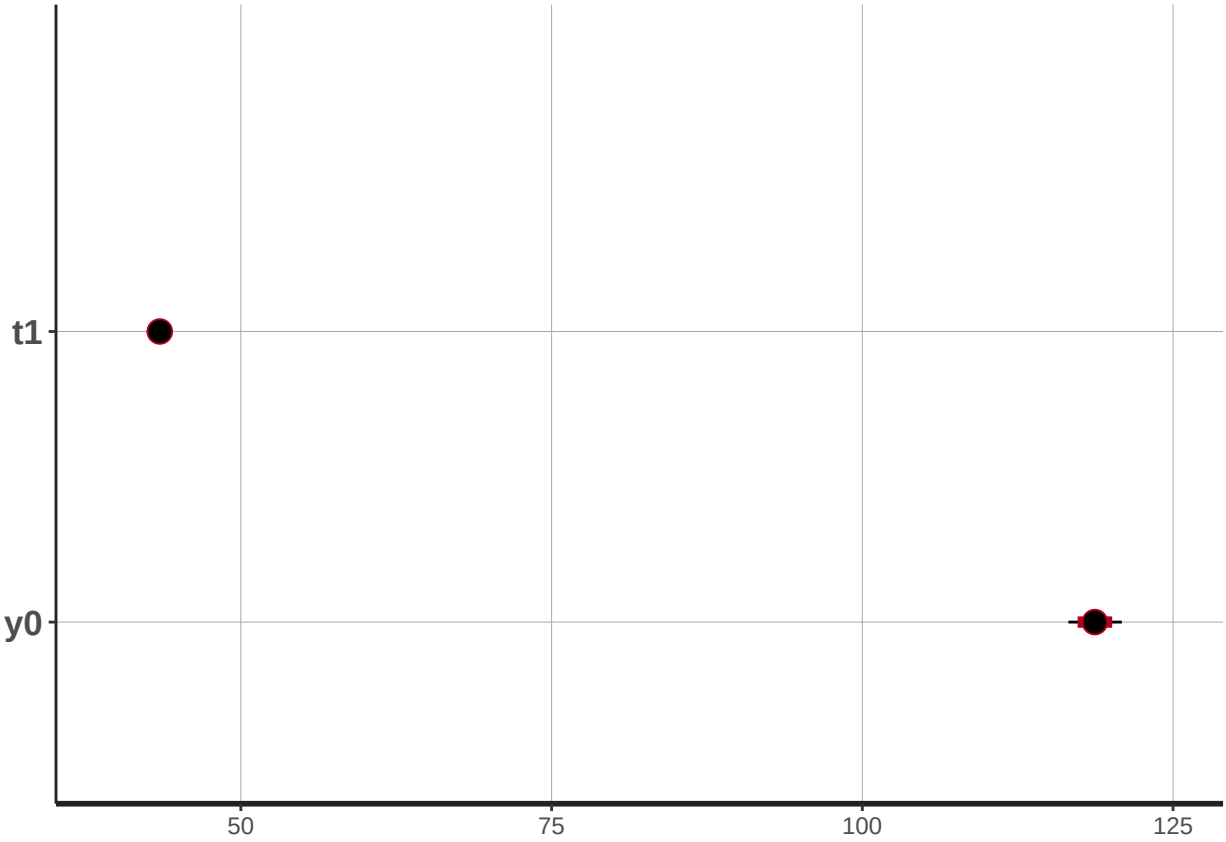
```
## Warning in ggplot2::geom_path(...): Ignoring unknown parameters: `main`
```



```
plot(fit_2, pars = pars, main = main)
```

```
## ci_level: 0.8 (80% intervals)
```

```
## outer_level: 0.95 (95% intervals)
```



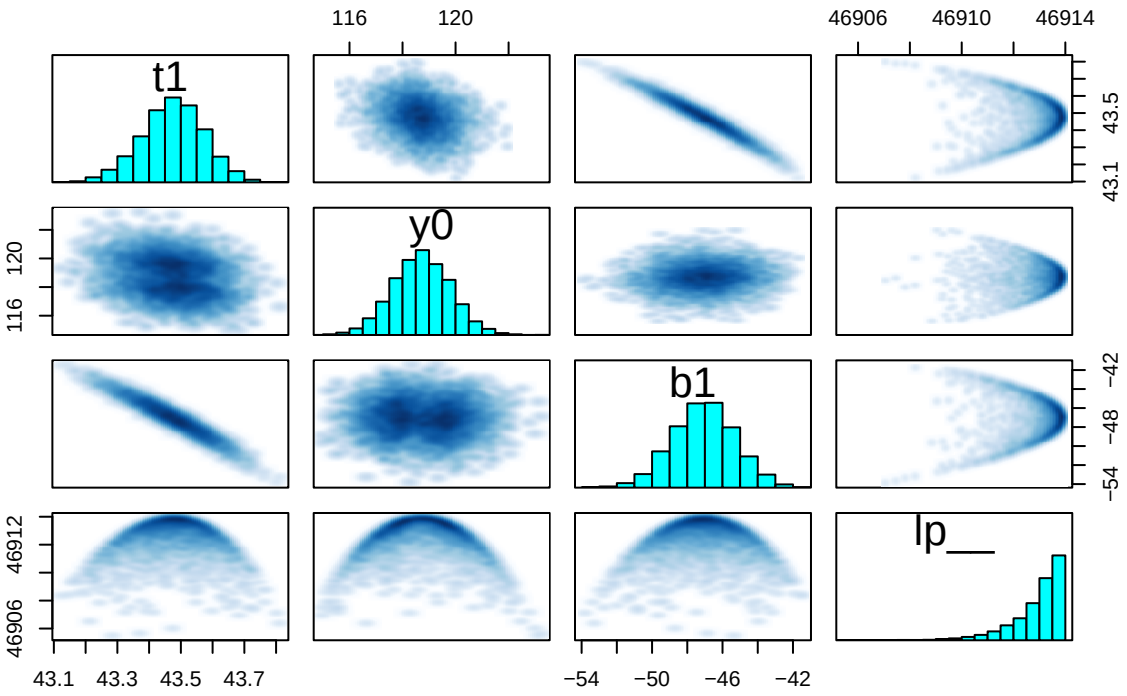

```
pairs(fit_2, pars = pars_full, main = main)
```

```
## Warning in par(usr): argument 1 does not name a graphical parameter
```

```
## Warning in par(usr): argument 1 does not name a graphical parameter
```

```
## Warning in par(usr): argument 1 does not name a graphical parameter
```

```
## Warning in par(usr): argument 1 does not name a graphical parameter
```




```
#ggplot()
```

Leave One Out (LOO) analysis

```
## Code from:
# Extract pointwise log-likelihood
# using merge_chains=FALSE returns an array, which is easier to
# use with relative_eff()
for(fit in c("fit_1", "fit_2")) {
  print(paste0("Model Fit: ", fit))
  log_lik_1 <- extract_log_lik(eval(parse(text = fit)), merge_chains = FALSE)
# as of loo v2.0.0 we can optionally provide relative effective sample sizes
# when calling loo, which allows for better estimates of the PSIS effective
# sample sizes and Monte Carlo error
r_eff <- relative_eff(exp(log_lik_1), cores = 2)

# preferably use more than 2 cores (as many cores as possible)
# will use value of 'mc.cores' option if cores is not specified
loo_1 <- loo(log_lik_1, r_eff = r_eff, cores = 2)
  print(loo_1)
}
```

```
## [1] "Model Fit: fit_1"
```

```
## Warning: Some Pareto k diagnostic values are too high. See help('pareto-k-diagnostic') for details.
```

```
##
```

```
## Computed from 10000 by 107 log-likelihood matrix
##
##          Estimate    SE
## elpd_loo  -3584.5 422.7
## p_loo        61.4  10.8
## looic      7168.9 845.4
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                            Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     101   94.4%   1022
##  (0.5, 0.7]   (ok)         4    3.7%   179
##    (0.7, 1]   (bad)        1    0.9%   41
##    (1, Inf)   (very bad)   1    0.9%   18
## See help('pareto-k-diagnostic') for details.
## [1] "Model Fit: fit_2"

## Warning: Some Pareto k diagnostic values are too high. See help('pareto-k-diagnostic') for details.

##
## Computed from 10000 by 107 log-likelihood matrix
##
##          Estimate    SE
## elpd_loo  -3550.8 417.4
## p_loo       126.0  35.0
## looic      7101.5 834.7
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                            Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     97    90.7%   730
##  (0.5, 0.7]   (ok)        2     1.9%   373
##    (0.7, 1]   (bad)       3     2.8%   14
##    (1, Inf)   (very bad)  5     4.7%   3
## See help('pareto-k-diagnostic') for details.
```

- Don't really know how to interpret these results.

Comparing Fits

Null Models: GLM_Poisson, GLM_Quasipoisson vs. STAN Poisson

```
summary(glm_poisson_0)
```

```
##
## Call:
## glm(formula = song_count ~ 1, family = poisson(link = "identity"),
##     data = data_ind)
##
```

```
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -15.264   -6.300   -2.558    5.042   21.979
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  116.495      1.043   111.6   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 6451.2  on 106  degrees of freedom
## Residual deviance: 6451.2  on 106  degrees of freedom
## AIC: 7107.5
##
## Number of Fisher Scoring iterations: 3
```

```
summary(glm_qpoisson_0)
```

```
##
## Call:
## glm(formula = song_count ~ 1, family = quasipoisson(link = "identity"),
##     data = data_ind)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -15.264   -6.300   -2.558    5.042   21.979
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  116.495      8.251   14.12   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasipoisson family taken to be 62.52277)
##
##     Null deviance: 6451.2  on 106  degrees of freedom
## Residual deviance: 6451.2  on 106  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 3
```

```
print(fit_1, pars = c("y0"))
```

```
## Inference for Stan model: one step poisson.
## 4 chains, each with iter=5000; warmup=2500; thin=1;
## post-warmup draws per chain=2500, total post-warmup draws=10000.
##
##      mean se_mean   sd   2.5%    25%    50%    75%  97.5% n_eff Rhat
## y0 116.48    0.01 1.04 114.47 115.78 116.48 117.18 118.51  8340    1
##
## Samples were drawn using NUTS(diag_e) at Fri Dec  2 14:49:30 2022.
```

```
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

- We see that all three approaches give the same best estimate of `y0 = 116.5`. In addition,
  - `poisson` and `stan` fits match in terms of their error estimate: 1.05. This is substantially smaller than the `quasipoisson` estimate of 8.25. I was hoping that the `stan` model would match the `quasipoisson` estimates.

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