foo

Anonymous

Contents

Introduction

Formula One (F1) is the most prestigious auto racing league in the world. The number of drivers in this racing class is only 20, and those taking part could be considered "the best of the best." Given the fierce level of competition, every attribute of the driver could be considered important. An F1 driver must be at least 18 years of age, that is, the participating racers are all adults. It is a well-known fact that motor skills and reflexes of adults deteriorate with age. In this report we explore the relationship between the

age and performance of F1 drivers through Bayesian analysis. In particular, we consider and compare three models: separate, pooled, and hierarchical. The models incorporate weakly priors, and related discussion of choices and sensitivity analysis is provided for each model separately. We further conduct posterior predictive analysis and finally provide a discussion on the models and potential future research.

Data

TODO: boxplot

Our analysis problem revolves around extracting posterior statistics yielding information about the relationship between a driver's age and performance. An example of such statistic would be the probability that the best age performance-wise lies within a certain range. The raw data is provided by the ("Ergast Developer API" n.d.). For the purposes of our analysis, we conduct an number of preprocessing steps on this dataset. We use only qualifying as the measure of performance and ignore race results in this analysis. In particular, we consider the difference between a driver and his teammate. In F1, a qualifying session consists of three timed periods, called Q1, Q2, and Q3, where 5 drivers are eliminated at the first two periods. From every qualifying for every driver we calculate the difference between a driver and his teammate from the last period they both participated in. To prune out outliers, we ignore cases where this difference is more than 2.5s, as this is rare and likely due to some unusual event unrelated to the driver's capability. We further subtract from this difference the historical mean value of the driver's difference to his teammates during his whole career to get comparable values for each driver that represent how they performed compared to their career average.

The same underlying data has been utilized by various authors. For example, (Nigro n.d.) examines how odds provided by a neural network model fit to a subset of the data fares against officially provided odds. Related to our analysis, the author notes that the age of the winning driver shows a decreasing linear trend. (Jain n.d.) on the other hand utilizes logistic regression with F_1 score to predict whether or not a driver manages to place on podium, i.e., to attain a top-three position. Similar to us, (van Kesteren n.d.) conducts a Bayesian analysis. The author models the skill of drivers considering factors including constructor advantages and seasonal constructor form. In contrast, our analysis is solely based on exploring whether or not a driver's age provides meaningful insight on his performance.

Models

In this section we enumerate and describe the three models we attempt to fit to the data described previously. Let us first define some general prerequisites. Let N denote the size of the dataset. Further, let $\mathcal{I} = \{1, 2, \ldots, N\}$ denote the index set of the dataset, and $\mathcal{A} = \{18, 19, \ldots, 43\}$ the set of driver ages present in the dataset. Now let $\text{age}(\cdot)$ be a mapping from $\mathcal{I}$ to $\mathcal{A}$ yielding the age of the sample at index $i \in \mathcal{I}$.

Separate

The separate model is formally defined as follows:

$$\begin{aligned} t_i &\sim \mathrm{N}(\mu_{\text{age}(i)}, \sigma_{\text{age}(i)}), \\ \mu_{\text{age}(i)} &\sim \mathrm{N}(0, 1), \\ \sigma_{\text{age}(i)} &\sim \mathrm{N}(0, 1). \end{aligned}$$

The separate model effectively assumes the defining characteristics of the distribution for t_i are different for each age group.

Pooled

The pooled model is formally defined as follows:

$$\begin{aligned} t_i &\sim \mathrm{N}(\mu, \sigma), \\ \mu &\sim \mathrm{N}(0, 1), \\ \sigma &\sim \mathrm{N}(0, 1). \end{aligned}$$

The pooled model simply assumes the characteristics of the distribution generating t_i are independent of driver age.

Hierarchical

The hierarchical model is defined as follows:

$$\begin{aligned} t_i &\sim \mathrm{N}(\mu_{\mathrm{age}(i)}, \sigma), \\ \mu_{\mathrm{age}(i)} &\sim \mathrm{N}(\mu_{\mathrm{unknown}}, \tau), \\ \mu_{\mathrm{unknown}} &\sim \mathrm{N}(0, 1), \\ \tau &\sim \mathrm{N}(0, 1), \\ \sigma &\sim \mathrm{N}(0, 1). \end{aligned}$$

In the hierarchical model, the variance of the distribution describing t_i is assumed the same for each age group. However, the mean of this distribution is assumed to be dependent on age. The parameters of the distribution yielding the mean of a particular age group are further sampled from the hyperpriors described above.

Separate with additional parameter

Additionally, we consider following separate model with additional parameter corresponding to teammate of the driver:

$$\begin{aligned} t_i &\sim \mathrm{N}(\mu_{\mathrm{age}(i)} + \alpha_{\mathrm{teammate(i)}}, \sigma_{\mathrm{age}(i)}), \\ \mu_{\mathrm{age}(i)} &\sim \mathrm{N}(0, 1), \\ \sigma_{\mathrm{age}(i)} &\sim \mathrm{N}(0, 1). \end{aligned}$$

Here, the α parameter is meant to capture the effect that teammate's level has to the expected difference for driver of certain age. If driver has a very strong teammate, his qualifying differences to the teammate are expected to be larger than usual and vice versa. We also briefly tested pooled and hierarchical model with the additional parameter but didn't see it as necessary to include the full analysis of those models to this report for since their comparison to each other is very similar to the comparison of the models without the parameter, and the comparison between one model without the parameter and one with is enough to see the effect of the parameter. Also, as mentioned in the convergence evaluation section, we were not able to achieve satisfying convergence with the hierarchical model with the additional parameter.

Weakly Prior Choices

In this kind of setting, we make a common assumption that the posterior distribution for t_i is of normal form. We refer to the data for prior information about the mean and variance of this distribution. After preprocessing, we have that the time differences between drivers lie within the interval $[-2.5, 2.5]$, which

gives rise to a weakly informative prior distribution for the mean of the distribution describing t_i as $\mathrm{N}(0, 1)$. Furthermore, it is evident that the differences between teammates is seldom over one second, which gives rise to a prior distribution for the variance of the distribution describing t_i also as $\mathrm{N}(0, 1)$. It turns out that the prior distributions are conjugate for the posterior distribution.

Stan Code

Separate model:

```
cat(readLines('../stan/separate_model.stan'), sep = '\n')
```

```
## // The input data N tells how many age pools we have
## // Total length tells how many rows in the input data
## // time contains the time differences to teammate
## // age contains age of driver at the time of the quali where the difference happened, where each row
## // corrseponds to to the row of time
## // model index is age transformed to corresponding model index (26 ages = 26 models)
## // rep ages contains the ages (models) for which replicated dataset is generated for predictive chec
## // rep_length is the length of the replicated dataset
## //
##
## functions {
##    // check if vector contains value
##    // if(r_in(3,{1,2,3,4})) will evaluate as 1
##   int r_in(int pos,int[] pos_var) {
##     for (p in 1:(size(pos_var))) {
##        if (pos_var[p]==pos) {
##           return 1;
##        }
##     }
##     return 0;
##   }
## }
##
## data {
##   int<lower=0> N;
##   int total_length;
##   real time[total_length];
##   int age[total_length];
##   int model_index[total_length];
##   int rep_ages[3];
##   int rep_length;
## }
##
## parameters {
##   real mu[N];
##   real<lower=0> sigma[N];
##
## }
##
##
##
```

```
## model {
##   mu ~ normal(0,1);        //prior
##   sigma ~ normal(0,1);     //prior
##   for(j in 1:total_length){
##     time[j] ~ normal(mu[model_index[j]], sigma[model_index[j]]);
##
##   }
## }
##
##
##
## generated quantities{
##   real yrep[rep_length];
##   real log_lik[total_length];
##   {
##   int  rep_index = 1;
##     for(j in 1:total_length){
##       if(r_in(age[j], rep_ages)) {
##         yrep[rep_index] = normal_rng(mu[model_index[j]], sigma[model_index[j]]);
##         rep_index += 1;
##       }
##     }
##   }
##
##   for(j in 1:total_length){
##     log_lik[j] =  normal_lpdf(time[j] | mu[model_index[j]], sigma[model_index[j]]);
##   }
## }
```

Pooled model:

```
cat(readLines('../stan/pooled_model.stan'), sep = '\n')
```

```
## // The input data N tells how many age pools we have
## // Total length tells how many rows in the input data
## // time contains the time differences to teammate
## // age contains age of driver at the time of the quali where the difference happened, where each row
## // corrseponds to to the row of time
## // rep ages contains the ages (models) for which replicated dataset is generated for predictive check
## // rep_length is the length of the replicated dataset
## //
## functions {
##    // check if vector contains value
##    // if(r_in(3,{1,2,3,4})) will evaluate as 1
##   int r_in(int pos,int[] pos_var) {
##     for (p in 1:(size(pos_var))) {
##        if (pos_var[p]==pos) {
##           return 1;
##        }
##     }
##     return 0;
##   }
## }
```

```
##
## data {
##   int<lower=0> N;
##   int total_length;
##   real time[total_length];
##   int age[total_length];
##   int rep_ages[3];
##   int rep_length;
## }
##
##
## parameters {
##   real mu;
##   real<lower=0> sigma;
## }
##
## model {
##   mu ~ normal(0,1);        //prior
##   sigma ~ normal(0,1);     //prior
##   time ~ normal(mu, sigma);
## }
##
## generated quantities{
##   real yrep[rep_length];
##   real log_lik[total_length];
##   {
##   int  rep_index = 1;
##     for(j in 1:total_length){
##       if(r_in(age[j], rep_ages)) {
##         yrep[rep_index] = normal_rng(mu, sigma);
##         rep_index += 1;
##       }
##     }
##   }
##   for(j in 1:total_length){
##     log_lik[j] =  normal_lpdf(time[j] | mu, sigma);
##   }
## }
```

Hierarchical model:

```
cat(readLines('../stan/hierarchical_model.stan'), sep = '\n')
```

```
## // The input data N tells how many age pools we have
## // Total length tells how many rows in the input data
## // time contains the time differences to teammate
## // age contains age of driver at the time of the quali where the difference happened, where each row
## // corrseponds to to the row of time
## // model index is age transformed to corresponding model index (26 ages = 26 models)
## // rep ages contains the ages (models) for which replicated dataset is generated for predictive chec
## // rep_length is the length of the replicated dataset
## //
##
```

```
## functions {
##    // if(r_in(3,{1,2,3,4})) will evaluate as 1
##   int r_in(int pos,int[] pos_var) {
##     for (p in 1:(size(pos_var))) {
##        if (pos_var[p]==pos) {
##           return 1;
##        }
##     }
##     return 0;
##   }
## }
##
## data {
##   int<lower=0> N;
##   int total_length;
##   real time[total_length];
##   int age[total_length];
##   int model_index[total_length];
##   int rep_ages[3];
##   int rep_length;
## }
##
##
## parameters {
##   real mu[N];
##   real<lower=0> sigma[N];
##   real<lower=0> tau;
##   real mu_mean;
##   real sigma_mean;
##   real<lower=0> sigma_tau;
## }
##
## model {
##   mu_mean ~ normal(0,1);    //hyperprior
##   tau ~ normal(0,1);       //hyperprior
##   mu ~ normal(mu_mean, tau);
##   sigma_mean ~ normal(0,1); //hyperprior
##   sigma_tau ~ normal(0,1); //hyperprior
##   sigma ~ normal(sigma_mean, sigma_tau);
##   for(j in 1:total_length){
##     time[j] ~ normal(mu[model_index[j]], sigma[model_index[j]]);
##   }
## }
##
## generated quantities{
##   real yrep[rep_length];
##   real log_lik[total_length];
##   {
##   int  rep_index = 1;
##     for(j in 1:total_length){
##       if(r_in(age[j], rep_ages)) {
##         yrep[rep_index] = normal_rng(mu[model_index[j]], sigma[model_index[j]]);
##         rep_index += 1;
##       }
```

```
##     }
##   }
##   for(j in 1:total_length){
##     log_lik[j] = normal_lpdf(time[j] | mu[model_index[j]], sigma[model_index[j]]);
##   }
## }
```

Separate model with additional parameter:

```
cat(readLines('../stan/separate_model_ids.stan'), sep = '\n')
```

```
## // The input data N tells how many age pools we have
## // Total length tells how many rows in the input data
## // time contains the time differences to teammate
## // age contains age of driver at the time of the quali where the difference happened, where each row
## // corrseponds to to the row of time
## // model index is age transformed to corresponding model index (26 ages = 26 models)
## // rep ages contains the ages (models) for which replicated dataset is generated for predictive check
## // rep_length is the length of the replicated dataset
## // driver_id contains driver_id's of teammates where each row corresponds to row of time
## // driver_count is the amount of drivers in the dataset
## //
##
## functions {
##    // if(r_in(3,{1,2,3,4})) will evaluate as 1
##   int r_in(int pos,int[] pos_var) {
##     for (p in 1:(size(pos_var))) {
##        if (pos_var[p]==pos) {
##           return 1;
##        }
##     }
##     return 0;
##   }
## }
##
## data {
##   int<lower=0> N;
##   int total_length;
##   real time[total_length];
##   int age[total_length];
##   int model_index[total_length];
##   int rep_ages[3];
##   int rep_length;
##   int driver_id[total_length];
##   int driver_count;
## }
##
##
## parameters {
##   real mu[N];
##   real<lower=0> sigma[N];
##   real a[driver_count];
## }
```

```
##
##
## model {
##   mu ~ normal(0,1);        //prior
##   sigma ~ normal(0,1);     //prior
##   a ~ normal(0,0.1);       //prior
##   for(j in 1:total_length){
##     time[j] ~ normal(mu[model_index[j]] + a[driver_id[j]], sigma[model_index[j]]);
##   }
## }
##
## generated quantities{
##   real yrep[rep_length];
##   real yrep_id[rep_length];      //replicated dataset where id is assumed to be known
##   real log_lik[total_length];
##   real log_lik_id[total_length]; //log likelihood where teammate is assumed to be known
##   {
##   int  rep_index = 1;
##     for(j in 1:total_length){
##       if(r_in(age[j], rep_ages)) {
##         yrep[rep_index] = normal_rng(mu[model_index[j]], sigma[model_index[j]]);
##         rep_index += 1;
##       }
##     }
##   }
##   {
##   int  rep_index = 1;
##     for(j in 1:total_length){
##       if(r_in(age[j], rep_ages)) {
##         yrep_id[rep_index] = normal_rng(mu[model_index[j]] + a[driver_id[j]], sigma[model_index[j]])
##         rep_index += 1;
##       }
##     }
##   }
##   for (j in 1:total_length){
##     log_lik[j] = normal_lpdf(time[j] | mu[model_index[j]], sigma[model_index[j]]);
## }
##   for (j in 1:total_length){
##     log_lik_id[j] = normal_lpdf(time[j] | mu[model_index[j]] + a[driver_id[j]], sigma[model_index[j]]
##  }
## }
```

Running the Stan Code

```
data = read.csv(file ="../../data/quali_differences_processed.csv")
data$teammateId = mapvalues(data$teammateId, unique(data$teammateId), 1:length(unique(data$teammateId)))
data$model_index = data$age-18+1


options(mc.cores = parallel::detectCores())
rstan_options(auto_write=TRUE)
```

```
ages = data$age
N = length(unique(ages))
y_rep_ages = c(19, 27, 41) #ages to generate replicated datasets for model evaluation
y_rep_length = sum(data$age %in% y_rep_ages)
y = data[data$age %in% y_rep_ages,]$difference

y_rep_groups = vector(mode="integer", length=y_rep_length)
rep_index = 1
for(j in 1:nrow(data)){
  if(ages[j] %in% y_rep_ages) {
    y_rep_groups[rep_index] = ages[j]
    rep_index = rep_index + 1
  }
}

stan_data <- list(N = N, total_length = nrow(data), time = data$difference, age = ages, min_age = min(ag
                  rep_ages = y_rep_ages, rep_length = y_rep_length)
stan_data_id <- list(N = N, total_length = nrow(data), time = data$difference, age = ages, min_age = min
                     rep_ages = y_rep_ages, rep_length = y_rep_length, driver_id = data$teammateId, driv
```

```
fit_separate <- stan(file = "../stan/separate_model.stan", data = stan_data)
fit_hierarchical <- stan(file = "../stan/hierarchical_model.stan", data = stan_data)
fit_pooled <- stan(file = "../stan/pooled_model.stan", data = stan_data)

fit_separate_id <- stan(file = "../stan/separate_model_ids.stan", data = stan_data_id, iter=6000)
```

Convergence Evaluation

According to (Vehtari et al. 2021), in the context of MCMC sampling, it is prudent to run at least four chains, which Stan performs by default. The authors further guide to accept the sample on the condition that $\widehat{R} < 1.05$, which we observe here.

Fit for each model showing $\widehat{R}$ and ESS:

```
#Separate
print(fit_separate, pars=c('mu', 'sigma'))
```

```
## Inference for Stan model: separate_model.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##            mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
## mu[1]      0.07       0 0.19 -0.32 -0.05  0.07  0.19  0.45  5832    1
## mu[2]      0.46       0 0.08  0.29  0.40  0.46  0.52  0.63  7547    1
## mu[3]     -0.01       0 0.05 -0.11 -0.04 -0.01  0.02  0.09  7246    1
## mu[4]      0.00       0 0.03 -0.06 -0.02  0.00  0.02  0.06  6589    1
## mu[5]      0.01       0 0.03 -0.04 -0.01  0.01  0.03  0.07  7382    1
## mu[6]      0.00       0 0.02 -0.04 -0.02  0.00  0.02  0.05  8572    1
## mu[7]     -0.09       0 0.03 -0.14 -0.10 -0.09 -0.07 -0.03  7227    1
## mu[8]     -0.06       0 0.03 -0.11 -0.08 -0.06 -0.04 -0.01  7930    1
## mu[9]     -0.08       0 0.02 -0.12 -0.09 -0.08 -0.06 -0.03  7092    1
```

```
## mu[10]    -0.16       0 0.03 -0.21 -0.18 -0.16 -0.14 -0.11  8044    1
## mu[11]    -0.05       0 0.03 -0.10 -0.06 -0.05 -0.03  0.01  7225    1
## mu[12]     0.00       0 0.03 -0.05 -0.02  0.00  0.02  0.05  7039    1
## mu[13]    -0.01       0 0.03 -0.06 -0.03 -0.01  0.01  0.05  7491    1
## mu[14]     0.02       0 0.03 -0.03  0.00  0.02  0.04  0.08  5779    1
## mu[15]     0.19       0 0.03  0.12  0.17  0.19  0.21  0.25  7788    1
## mu[16]     0.08       0 0.04  0.01  0.06  0.08  0.10  0.15  8543    1
## mu[17]     0.07       0 0.04  0.00  0.05  0.07  0.09  0.14  8106    1
## mu[18]     0.13       0 0.04  0.06  0.10  0.13  0.15  0.20  6156    1
## mu[19]     0.01       0 0.04 -0.07 -0.02  0.01  0.04  0.09  6288    1
## mu[20]     0.15       0 0.04  0.07  0.12  0.15  0.18  0.23  7774    1
## mu[21]     0.14       0 0.06  0.02  0.10  0.14  0.18  0.25  7640    1
## mu[22]     0.12       0 0.06  0.01  0.08  0.12  0.15  0.23  6370    1
## mu[23]     0.07       0 0.09 -0.10  0.01  0.07  0.12  0.24  6777    1
## mu[24]     0.04       0 0.09 -0.14 -0.02  0.04  0.10  0.22  7530    1
## mu[25]     0.46       0 0.12  0.22  0.37  0.46  0.54  0.70  6374    1
## mu[26]     0.31       0 0.15  0.02  0.22  0.31  0.40  0.60  5420    1
## sigma[1]   0.79       0 0.15  0.55  0.68  0.76  0.87  1.14  4143    1
## sigma[2]   0.59       0 0.06  0.48  0.55  0.58  0.62  0.72  7642    1
## sigma[3]   0.57       0 0.04  0.50  0.54  0.57  0.59  0.64  8023    1
## sigma[4]   0.56       0 0.02  0.52  0.54  0.56  0.58  0.61  6526    1
## sigma[5]   0.59       0 0.02  0.55  0.58  0.59  0.60  0.63  7448    1
## sigma[6]   0.59       0 0.02  0.56  0.58  0.59  0.60  0.62  7734    1
## sigma[7]   0.64       0 0.02  0.61  0.63  0.64  0.65  0.68  6693    1
## sigma[8]   0.65       0 0.02  0.62  0.64  0.65  0.66  0.69  7059    1
## sigma[9]   0.63       0 0.02  0.60  0.62  0.63  0.64  0.66  7861    1
## sigma[10]  0.63       0 0.02  0.59  0.62  0.63  0.64  0.67  6913    1
## sigma[11]  0.67       0 0.02  0.63  0.66  0.67  0.68  0.71  5749    1
## sigma[12]  0.61       0 0.02  0.57  0.59  0.61  0.62  0.64  7612    1
## sigma[13]  0.64       0 0.02  0.60  0.62  0.64  0.65  0.67  7857    1
## sigma[14]  0.67       0 0.02  0.63  0.65  0.67  0.68  0.71  5953    1
## sigma[15]  0.61       0 0.02  0.57  0.59  0.61  0.62  0.65  7564    1
## sigma[16]  0.63       0 0.03  0.58  0.61  0.62  0.64  0.68  6827    1
## sigma[17]  0.65       0 0.03  0.60  0.63  0.65  0.67  0.70  7683    1
## sigma[18]  0.63       0 0.02  0.59  0.62  0.63  0.65  0.68  7906    1
## sigma[19]  0.67       0 0.03  0.61  0.65  0.67  0.69  0.73  6190    1
## sigma[20]  0.54       0 0.03  0.49  0.52  0.54  0.56  0.60  7015    1
## sigma[21]  0.54       0 0.04  0.47  0.51  0.54  0.57  0.63  7221    1
## sigma[22]  0.39       0 0.04  0.32  0.36  0.39  0.42  0.48  6006    1
## sigma[23]  0.54       0 0.06  0.43  0.49  0.53  0.58  0.68  5573    1
## sigma[24]  0.67       0 0.06  0.56  0.62  0.66  0.71  0.81  6333    1
## sigma[25]  0.74       0 0.09  0.59  0.68  0.73  0.80  0.94  6181    1
## sigma[26]  0.63       0 0.11  0.46  0.56  0.62  0.69  0.91  4476    1
##
## Samples were drawn using NUTS(diag_e) at Mon Dec 06 20:29:06 2021.
## 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).
```

```
#hierarchical
print(fit_hierarchical, pars=c('mu', 'sigma'))
```

```
## Inference for Stan model: hierarchical_model.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
```

```
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##             mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
## mu[1]       0.06       0 0.10 -0.12 -0.01  0.06  0.12  0.25  7640    1
## mu[2]       0.32       0 0.08  0.17  0.26  0.32  0.37  0.49  5248    1
## mu[3]       0.00       0 0.05 -0.10 -0.03  0.00  0.04  0.10  7444    1
## mu[4]       0.00       0 0.03 -0.06 -0.02  0.00  0.03  0.07  7953    1
## mu[5]       0.02       0 0.03 -0.04  0.00  0.02  0.03  0.07  8485    1
## mu[6]       0.00       0 0.02 -0.04 -0.01  0.00  0.02  0.05  7660    1
## mu[7]      -0.08       0 0.03 -0.13 -0.10 -0.08 -0.06 -0.03  8412    1
## mu[8]      -0.06       0 0.02 -0.10 -0.07 -0.06 -0.04 -0.01  7714    1
## mu[9]      -0.07       0 0.02 -0.11 -0.09 -0.07 -0.06 -0.03  6638    1
## mu[10]     -0.15       0 0.02 -0.20 -0.16 -0.15 -0.13 -0.10  7992    1
## mu[11]     -0.04       0 0.03 -0.09 -0.06 -0.04 -0.03  0.01  7868    1
## mu[12]      0.00       0 0.03 -0.05 -0.02  0.00  0.02  0.05  8685    1
## mu[13]     -0.01       0 0.03 -0.05 -0.02  0.00  0.01  0.04  7851    1
## mu[14]      0.02       0 0.03 -0.03  0.01  0.03  0.04  0.08  8165    1
## mu[15]      0.18       0 0.03  0.12  0.16  0.18  0.20  0.24  7440    1
## mu[16]      0.08       0 0.03  0.01  0.06  0.08  0.10  0.14  8524    1
## mu[17]      0.07       0 0.03  0.00  0.05  0.07  0.09  0.14  7906    1
## mu[18]      0.12       0 0.03  0.06  0.10  0.12  0.14  0.19  8257    1
## mu[19]      0.01       0 0.04 -0.06 -0.01  0.02  0.04  0.09  8392    1
## mu[20]      0.14       0 0.04  0.06  0.11  0.14  0.16  0.22  7942    1
## mu[21]      0.12       0 0.06  0.01  0.08  0.12  0.16  0.23  7847    1
## mu[22]      0.10       0 0.06 -0.02  0.06  0.10  0.14  0.22  7221    1
## mu[23]      0.06       0 0.07 -0.08  0.01  0.06  0.11  0.21  8579    1
## mu[24]      0.05       0 0.07 -0.09  0.00  0.05  0.09  0.19  8557    1
## mu[25]      0.28       0 0.09  0.10  0.22  0.28  0.34  0.46  5299    1
## mu[26]      0.16       0 0.10 -0.01  0.10  0.16  0.22  0.36  5623    1
## sigma[1]    0.63       0 0.04  0.56  0.61  0.63  0.65  0.71  6682    1
## sigma[2]    0.61       0 0.03  0.54  0.59  0.61  0.63  0.68  6530    1
## sigma[3]    0.59       0 0.03  0.54  0.57  0.59  0.61  0.65  4807    1
## sigma[4]    0.58       0 0.02  0.54  0.56  0.58  0.59  0.62  3344    1
## sigma[5]    0.60       0 0.02  0.56  0.58  0.60  0.61  0.63  5686    1
## sigma[6]    0.59       0 0.02  0.56  0.58  0.59  0.60  0.62  5506    1
## sigma[7]    0.64       0 0.02  0.61  0.63  0.64  0.65  0.67  7292    1
## sigma[8]    0.64       0 0.02  0.61  0.63  0.64  0.65  0.68  6912    1
## sigma[9]    0.63       0 0.01  0.60  0.62  0.63  0.64  0.66  8311    1
## sigma[10]   0.63       0 0.02  0.60  0.61  0.63  0.64  0.66  7239    1
## sigma[11]   0.66       0 0.02  0.63  0.65  0.66  0.67  0.69  4457    1
## sigma[12]   0.61       0 0.02  0.58  0.60  0.61  0.62  0.64 10582    1
## sigma[13]   0.63       0 0.02  0.60  0.62  0.63  0.64  0.67  8548    1
## sigma[14]   0.65       0 0.02  0.62  0.64  0.65  0.67  0.69  5280    1
## sigma[15]   0.61       0 0.02  0.58  0.60  0.61  0.62  0.65  7813    1
## sigma[16]   0.62       0 0.02  0.59  0.61  0.62  0.63  0.66  7860    1
## sigma[17]   0.64       0 0.02  0.60  0.62  0.64  0.65  0.68  6617    1
## sigma[18]   0.63       0 0.02  0.59  0.61  0.63  0.64  0.67  8480    1
## sigma[19]   0.65       0 0.02  0.61  0.63  0.65  0.66  0.69  5373    1
## sigma[20]   0.57       0 0.03  0.52  0.55  0.57  0.59  0.62  2854    1
## sigma[21]   0.59       0 0.03  0.52  0.57  0.59  0.61  0.65  4145    1
## sigma[22]   0.55       0 0.05  0.44  0.51  0.55  0.59  0.63  1758    1
## sigma[23]   0.60       0 0.04  0.52  0.57  0.60  0.62  0.66  4049    1
## sigma[24]   0.63       0 0.03  0.57  0.61  0.63  0.65  0.70  8064    1
## sigma[25]   0.64       0 0.04  0.58  0.62  0.64  0.67  0.73  4946    1
```

```
## sigma[26]  0.62       0 0.04  0.54  0.59  0.62  0.64  0.69  6406    1
##
## Samples were drawn using NUTS(diag_e) at Mon Dec 06 20:30:20 2021.
## 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).
```

```
#pooled
print(fit_pooled, pars=c('mu', 'sigma'))
```

```
## Inference for Stan model: pooled_model.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##       mean se_mean   sd  2.5%  25%  50%  75% 97.5% n_eff Rhat
## mu    0.00       0 0.01 -0.01 0.00 0.00 0.00  0.01  4019    1
## sigma 0.63       0 0.00  0.62 0.63 0.63 0.63  0.64  3951    1
##
## Samples were drawn using NUTS(diag_e) at Mon Dec 06 20:31:15 2021.
## 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).
```

```
#Separate_id (separate with additional parameter)
print(fit_separate_id, pars=c('mu', 'sigma', 'a'))
```

```
## Inference for Stan model: separate_model_ids.
## 4 chains, each with iter=6000; warmup=3000; thin=1;
## post-warmup draws per chain=3000, total post-warmup draws=12000.
##
##            mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
## mu[1]     -0.06       0 0.20 -0.45 -0.18 -0.06  0.07  0.35 20703    1
## mu[2]      0.39       0 0.09  0.21  0.33  0.39  0.45  0.57 22435    1
## mu[3]     -0.03       0 0.06 -0.14 -0.07 -0.03  0.01  0.08 21260    1
## mu[4]     -0.08       0 0.04 -0.15 -0.10 -0.08 -0.05  0.00 17742    1
## mu[5]     -0.06       0 0.03 -0.12 -0.08 -0.06 -0.04  0.00 16737    1
## mu[6]     -0.05       0 0.03 -0.10 -0.07 -0.05 -0.04  0.00 15920    1
## mu[7]     -0.14       0 0.03 -0.19 -0.16 -0.14 -0.12 -0.08 16951    1
## mu[8]     -0.15       0 0.03 -0.20 -0.16 -0.14 -0.13 -0.09 15923    1
## mu[9]     -0.15       0 0.03 -0.20 -0.17 -0.15 -0.13 -0.10 14857    1
## mu[10]    -0.22       0 0.03 -0.28 -0.24 -0.22 -0.20 -0.17 16114    1
## mu[11]    -0.11       0 0.03 -0.17 -0.13 -0.11 -0.09 -0.06 16947    1
## mu[12]    -0.06       0 0.03 -0.12 -0.08 -0.06 -0.04 -0.01 17177    1
## mu[13]    -0.10       0 0.03 -0.16 -0.12 -0.10 -0.08 -0.05 18517    1
## mu[14]    -0.07       0 0.03 -0.13 -0.09 -0.07 -0.05 -0.01 17397    1
## mu[15]     0.06       0 0.03 -0.01  0.03  0.06  0.08  0.12 18501    1
## mu[16]    -0.02       0 0.04 -0.09 -0.04 -0.02  0.01  0.06 18572    1
## mu[17]     0.04       0 0.04 -0.04  0.01  0.04  0.06  0.11 20046    1
## mu[18]     0.05       0 0.04 -0.02  0.03  0.05  0.08  0.13 18069    1
## mu[19]    -0.05       0 0.04 -0.14 -0.08 -0.05 -0.03  0.03 18406    1
## mu[20]     0.09       0 0.04  0.01  0.06  0.09  0.12  0.18 20379    1
## mu[21]     0.06       0 0.07 -0.07  0.01  0.06  0.10  0.19 21772    1
## mu[22]     0.03       0 0.06 -0.09 -0.01  0.03  0.07  0.15 18634    1
```

```
## mu[23]     0.06       0 0.09 -0.13 -0.01  0.06  0.12  0.24 20538    1
## mu[24]     0.05       0 0.09 -0.12  0.00  0.05  0.11  0.22 20961    1
## mu[25]     0.38       0 0.13  0.13  0.30  0.38  0.47  0.63 22675    1
## mu[26]     0.14       0 0.15 -0.17  0.04  0.13  0.24  0.44 20721    1
## sigma[1]   0.78       0 0.15  0.55  0.68  0.76  0.87  1.13 19780    1
## sigma[2]   0.60       0 0.06  0.49  0.56  0.59  0.64  0.74 22099    1
## sigma[3]   0.56       0 0.04  0.49  0.53  0.56  0.58  0.63 26148    1
## sigma[4]   0.55       0 0.02  0.50  0.53  0.55  0.57  0.60 26884    1
## sigma[5]   0.57       0 0.02  0.54  0.56  0.57  0.58  0.61 26622    1
## sigma[6]   0.58       0 0.02  0.55  0.57  0.58  0.59  0.61 28360    1
## sigma[7]   0.63       0 0.02  0.60  0.62  0.63  0.64  0.67 27844    1
## sigma[8]   0.64       0 0.02  0.60  0.63  0.64  0.65  0.68 30130    1
## sigma[9]   0.61       0 0.02  0.58  0.60  0.61  0.62  0.64 28311    1
## sigma[10]  0.60       0 0.02  0.57  0.59  0.60  0.62  0.64 27511    1
## sigma[11]  0.65       0 0.02  0.62  0.64  0.65  0.66  0.69 29944    1
## sigma[12]  0.59       0 0.02  0.55  0.58  0.59  0.60  0.62 27665    1
## sigma[13]  0.62       0 0.02  0.58  0.60  0.62  0.63  0.65 27522    1
## sigma[14]  0.63       0 0.02  0.59  0.62  0.63  0.64  0.67 26531    1
## sigma[15]  0.59       0 0.02  0.55  0.58  0.59  0.61  0.63 26933    1
## sigma[16]  0.61       0 0.02  0.56  0.59  0.61  0.62  0.66 27860    1
## sigma[17]  0.65       0 0.03  0.60  0.63  0.65  0.67  0.71 27598    1
## sigma[18]  0.61       0 0.02  0.56  0.59  0.61  0.63  0.66 30361    1
## sigma[19]  0.64       0 0.03  0.59  0.62  0.64  0.66  0.70 28060    1
## sigma[20]  0.52       0 0.03  0.47  0.50  0.52  0.54  0.58 29242    1
## sigma[21]  0.55       0 0.05  0.47  0.52  0.55  0.58  0.65 26069    1
## sigma[22]  0.38       0 0.04  0.31  0.35  0.38  0.40  0.46 25810    1
## sigma[23]  0.53       0 0.06  0.43  0.49  0.53  0.57  0.67 23512    1
## sigma[24]  0.58       0 0.06  0.48  0.54  0.58  0.62  0.71 25692    1
## sigma[25]  0.72       0 0.09  0.57  0.65  0.71  0.77  0.92 21679    1
## sigma[26]  0.64       0 0.11  0.46  0.56  0.62  0.70  0.91 16892    1
## a[1]       0.07       0 0.05 -0.03  0.04  0.07  0.11  0.18 23958    1
## a[2]       0.11       0 0.05  0.02  0.08  0.11  0.14  0.20 22550    1
## a[3]       0.26       0 0.04  0.19  0.24  0.26  0.29  0.33 20202    1
## a[4]       0.12       0 0.04  0.05  0.10  0.12  0.15  0.19 18984    1
## a[5]       0.12       0 0.05  0.01  0.08  0.12  0.15  0.22 24086    1
## a[6]       0.02       0 0.06 -0.09 -0.01  0.02  0.06  0.13 26511    1
## a[7]      -0.09       0 0.07 -0.23 -0.14 -0.09 -0.04  0.04 28291    1
## a[8]       0.12       0 0.04  0.04  0.09  0.12  0.15  0.20 22819    1
## a[9]       0.20       0 0.04  0.11  0.17  0.20  0.23  0.28 25059    1
## a[10]     -0.05       0 0.07 -0.19 -0.10 -0.05  0.00  0.10 28704    1
## a[11]      0.10       0 0.04  0.03  0.08  0.10  0.13  0.17 21456    1
## a[12]     -0.03       0 0.08 -0.19 -0.09 -0.03  0.02  0.12 29596    1
## a[13]      0.09       0 0.04  0.02  0.07  0.09  0.12  0.17 22236    1
## a[14]      0.17       0 0.05  0.09  0.14  0.18  0.21  0.26 21012    1
## a[15]     -0.03       0 0.05 -0.12 -0.06 -0.03  0.00  0.06 24581    1
## a[16]      0.03       0 0.04 -0.05  0.00  0.03  0.06  0.11 18366    1
## a[17]      0.14       0 0.05  0.04  0.10  0.14  0.17  0.24 27975    1
## a[18]      0.17       0 0.04  0.10  0.15  0.17  0.20  0.25 18460    1
## a[19]      0.13       0 0.05  0.03  0.10  0.13  0.16  0.22 22993    1
## a[20]      0.06       0 0.08 -0.10  0.00  0.06  0.11  0.22 33941    1
## a[21]      0.21       0 0.04  0.14  0.18  0.21  0.23  0.28 20940    1
## a[22]     -0.05       0 0.06 -0.17 -0.09 -0.05 -0.01  0.08 29580    1
## a[23]      0.06       0 0.05 -0.05  0.02  0.06  0.09  0.16 27374    1
## a[24]     -0.09       0 0.07 -0.23 -0.14 -0.09 -0.04  0.05 27815    1
```

```
## a[25]      0.01       0 0.06 -0.10 -0.03  0.01  0.05  0.13 25522    1
## a[26]     -0.05       0 0.08 -0.20 -0.11 -0.05  0.00  0.10 30406    1
## a[27]      0.12       0 0.07 -0.02  0.08  0.12  0.17  0.26 30545    1
## a[28]     -0.03       0 0.10 -0.22 -0.09 -0.03  0.04  0.17 33427    1
## a[29]     -0.15       0 0.08 -0.31 -0.21 -0.15 -0.10  0.01 28881    1
## a[30]      0.34       0 0.04  0.25  0.31  0.34  0.37  0.43 22839    1
## a[31]      0.10       0 0.06 -0.01  0.06  0.10  0.14  0.22 26327    1
## a[32]     -0.14       0 0.07 -0.28 -0.18 -0.14 -0.09  0.00 28979    1
## a[33]      0.03       0 0.08 -0.12 -0.02  0.03  0.08  0.18 27774    1
## a[34]      0.01       0 0.06 -0.11 -0.03  0.01  0.06  0.14 29055    1
## a[35]     -0.10       0 0.10 -0.29 -0.16 -0.10 -0.03  0.09 26714    1
## a[36]     -0.06       0 0.09 -0.25 -0.13 -0.06  0.00  0.12 31535    1
## a[37]      0.03       0 0.07 -0.10 -0.02  0.03  0.07  0.16 29298    1
## a[38]      0.01       0 0.09 -0.17 -0.06  0.00  0.07  0.18 33608    1
## a[39]     -0.22       0 0.07 -0.37 -0.27 -0.22 -0.17 -0.07 28369    1
## a[40]      0.00       0 0.09 -0.18 -0.06  0.00  0.06  0.17 29854    1
## a[41]     -0.11       0 0.09 -0.29 -0.17 -0.11 -0.05  0.06 30030    1
## a[42]     -0.18       0 0.08 -0.34 -0.24 -0.18 -0.13 -0.02 30163    1
## a[43]      0.03       0 0.06 -0.09 -0.01  0.03  0.07  0.14 30282    1
## a[44]     -0.08       0 0.09 -0.25 -0.13 -0.08 -0.02  0.10 30400    1
## a[45]      0.01       0 0.09 -0.16 -0.04  0.01  0.07  0.18 30126    1
## a[46]      0.01       0 0.09 -0.16 -0.04  0.01  0.07  0.18 28183    1
## a[47]      0.08       0 0.06 -0.03  0.04  0.08  0.12  0.19 26581    1
## a[48]     -0.01       0 0.07 -0.16 -0.06 -0.02  0.03  0.13 26935    1
## a[49]     -0.19       0 0.09 -0.35 -0.24 -0.19 -0.13 -0.02 30421    1
## a[50]     -0.06       0 0.09 -0.23 -0.12 -0.06 -0.01  0.10 28920    1
## a[51]     -0.06       0 0.07 -0.19 -0.10 -0.06 -0.01  0.07 29150    1
## a[52]     -0.05       0 0.10 -0.24 -0.11 -0.05  0.02  0.15 32769    1
## a[53]      0.01       0 0.10 -0.18 -0.06  0.01  0.07  0.20 30469    1
## a[54]      0.04       0 0.06 -0.08  0.00  0.04  0.08  0.16 28527    1
## a[55]      0.09       0 0.07 -0.05  0.04  0.09  0.13  0.22 27229    1
## a[56]     -0.04       0 0.10 -0.23 -0.11 -0.04  0.02  0.15 31550    1
## a[57]      0.12       0 0.06  0.00  0.08  0.12  0.17  0.25 27381    1
## a[58]      0.00       0 0.06 -0.12 -0.04  0.00  0.04  0.12 26637    1
## a[59]     -0.01       0 0.10 -0.21 -0.08 -0.01  0.06  0.18 31769    1
## a[60]     -0.15       0 0.07 -0.27 -0.19 -0.15 -0.10 -0.02 33690    1
## a[61]     -0.14       0 0.07 -0.27 -0.18 -0.14 -0.09  0.00 28220    1
## a[62]     -0.06       0 0.09 -0.24 -0.12 -0.06  0.01  0.13 27896    1
## a[63]     -0.04       0 0.09 -0.21 -0.10 -0.04  0.02  0.13 31224    1
## a[64]      0.13       0 0.06  0.01  0.09  0.13  0.17  0.26 27021    1
## a[65]     -0.08       0 0.09 -0.25 -0.14 -0.08 -0.02  0.09 30823    1
## a[66]      0.00       0 0.09 -0.18 -0.06  0.00  0.06  0.18 28106    1
## a[67]      0.02       0 0.08 -0.13 -0.03  0.02  0.07  0.17 27657    1
## a[68]      0.02       0 0.10 -0.18 -0.05  0.02  0.09  0.22 31504    1
## a[69]     -0.18       0 0.09 -0.35 -0.24 -0.18 -0.12 -0.01 30920    1
## a[70]     -0.22       0 0.09 -0.40 -0.28 -0.22 -0.16 -0.05 29741    1
## a[71]     -0.11       0 0.08 -0.27 -0.16 -0.11 -0.05  0.06 27271    1
## a[72]     -0.05       0 0.09 -0.23 -0.11 -0.05  0.02  0.14 29552    1
## a[73]     -0.17       0 0.09 -0.34 -0.23 -0.16 -0.10  0.01 30889    1
## a[74]      0.04       0 0.07 -0.09 -0.01  0.04  0.08  0.17 24940    1
## a[75]     -0.04       0 0.07 -0.17 -0.08 -0.04  0.01  0.09 27115    1
## a[76]      0.00       0 0.08 -0.16 -0.06 -0.01  0.05  0.15 30736    1
## a[77]     -0.07       0 0.10 -0.27 -0.14 -0.07 -0.01  0.12 26885    1
## a[78]      0.00       0 0.10 -0.19 -0.07  0.00  0.06  0.19 32418    1
```

```
## a[79]      0.02       0 0.07 -0.13 -0.03  0.02  0.07  0.16 26170    1
## a[80]     -0.01       0 0.08 -0.17 -0.06 -0.01  0.05  0.16 30759    1
## a[81]      0.02       0 0.08 -0.14 -0.03  0.02  0.08  0.19 31402    1
## a[82]     -0.11       0 0.08 -0.26 -0.16 -0.11 -0.05  0.05 29548    1
## a[83]     -0.14       0 0.09 -0.32 -0.20 -0.14 -0.08  0.04 31916    1
## a[84]     -0.08       0 0.09 -0.26 -0.15 -0.08 -0.02  0.09 29331    1
## a[85]      0.00       0 0.08 -0.16 -0.05  0.00  0.06  0.16 26773    1
## a[86]      0.03       0 0.09 -0.14 -0.03  0.03  0.09  0.21 31414    1
## a[87]     -0.12       0 0.09 -0.30 -0.18 -0.12 -0.06  0.06 30460    1
## a[88]     -0.04       0 0.09 -0.21 -0.10 -0.04  0.02  0.13 30885    1
## a[89]      0.02       0 0.09 -0.17 -0.05  0.02  0.08  0.20 29896    1
## a[90]     -0.02       0 0.09 -0.19 -0.08 -0.02  0.04  0.16 33392    1
## a[91]     -0.16       0 0.09 -0.34 -0.22 -0.16 -0.10  0.01 36529    1
## a[92]     -0.04       0 0.09 -0.22 -0.10 -0.04  0.02  0.15 29123    1
## a[93]     -0.08       0 0.10 -0.27 -0.15 -0.08 -0.02  0.11 29087    1
## a[94]     -0.02       0 0.10 -0.22 -0.09 -0.02  0.04  0.17 29382    1
## a[95]      0.07       0 0.10 -0.12  0.01  0.07  0.14  0.26 28416    1
## a[96]      0.05       0 0.09 -0.12 -0.01  0.05  0.11  0.22 31057    1
## a[97]     -0.04       0 0.09 -0.22 -0.10 -0.04  0.03  0.14 29949    1
## a[98]     -0.02       0 0.09 -0.19 -0.08 -0.02  0.03  0.14 31931    1
## a[99]     -0.01       0 0.09 -0.18 -0.07  0.00  0.05  0.17 29758    1
## a[100]     0.03       0 0.09 -0.13 -0.03  0.03  0.09  0.20 32506    1
## a[101]    -0.01       0 0.10 -0.20 -0.08 -0.01  0.06  0.18 33241    1
## a[102]     0.03       0 0.09 -0.15 -0.03  0.03  0.09  0.20 30343    1
## a[103]    -0.06       0 0.09 -0.25 -0.13 -0.06  0.00  0.12 34669    1
## a[104]    -0.08       0 0.10 -0.27 -0.14 -0.08 -0.01  0.12 29755    1
## a[105]    -0.04       0 0.09 -0.21 -0.10 -0.04  0.02  0.14 29641    1
## a[106]     0.04       0 0.09 -0.14 -0.02  0.04  0.10  0.22 27068    1
## a[107]     0.01       0 0.10 -0.18 -0.06  0.01  0.07  0.20 29871    1
## a[108]     0.02       0 0.10 -0.17 -0.05  0.02  0.09  0.21 33230    1
## a[109]    -0.05       0 0.10 -0.25 -0.12 -0.05  0.01  0.14 30233    1
## a[110]     0.00       0 0.10 -0.19 -0.06  0.00  0.07  0.20 32506    1
## a[111]    -0.02       0 0.10 -0.21 -0.08 -0.02  0.05  0.17 29971    1
## a[112]    -0.04       0 0.10 -0.24 -0.11 -0.04  0.03  0.16 30413    1
## a[113]    -0.01       0 0.06 -0.14 -0.05 -0.01  0.03  0.11 26735    1
## a[114]     0.03       0 0.07 -0.09 -0.01  0.03  0.08  0.16 28412    1
## a[115]     0.13       0 0.04  0.05  0.10  0.13  0.16  0.22 22307    1
## a[116]     0.10       0 0.06 -0.01  0.07  0.10  0.14  0.22 22556    1
## a[117]    -0.12       0 0.07 -0.25 -0.17 -0.12 -0.08  0.00 29434    1
## a[118]     0.17       0 0.04  0.08  0.14  0.17  0.20  0.25 22680    1
## a[119]    -0.09       0 0.09 -0.26 -0.15 -0.09 -0.03  0.08 30855    1
## a[120]    -0.06       0 0.09 -0.23 -0.12 -0.06  0.00  0.11 33585    1
## a[121]    -0.11       0 0.07 -0.25 -0.16 -0.11 -0.07  0.02 27711    1
## a[122]    -0.12       0 0.10 -0.31 -0.19 -0.12 -0.06  0.07 31575    1
## a[123]     0.06       0 0.04 -0.02  0.03  0.06  0.08  0.14 20455    1
## a[124]    -0.02       0 0.05 -0.12 -0.05 -0.02  0.02  0.09 23262    1
## a[125]    -0.12       0 0.08 -0.29 -0.18 -0.12 -0.07  0.04 26549    1
## a[126]     0.10       0 0.06 -0.03  0.06  0.10  0.15  0.23 27410    1
## a[127]     0.15       0 0.04  0.06  0.12  0.15  0.17  0.23 21273    1
## a[128]    -0.07       0 0.06 -0.19 -0.11 -0.07 -0.02  0.06 28040    1
## a[129]     0.01       0 0.07 -0.13 -0.04  0.01  0.06  0.16 27420    1
## a[130]    -0.05       0 0.06 -0.18 -0.09 -0.05 -0.01  0.08 27795    1
## a[131]     0.02       0 0.05 -0.06 -0.01  0.02  0.06  0.12 22045    1
## a[132]     0.06       0 0.08 -0.09  0.01  0.06  0.11  0.21 30639    1
```

```
## a[133]     0.03       0 0.07 -0.12 -0.02  0.03  0.07  0.17 27925    1
## a[134]    -0.07       0 0.08 -0.23 -0.12 -0.07 -0.02  0.09 32387    1
## a[135]    -0.05       0 0.05 -0.15 -0.09 -0.05 -0.02  0.05 26425    1
## a[136]     0.02       0 0.05 -0.08 -0.02  0.02  0.05  0.12 26675    1
## a[137]    -0.04       0 0.05 -0.15 -0.08 -0.04  0.00  0.07 26069    1
## a[138]     0.02       0 0.10 -0.18 -0.05  0.02  0.09  0.22 29504    1
## a[139]     0.03       0 0.08 -0.13 -0.02  0.03  0.09  0.20 30766    1
## a[140]     0.25       0 0.05  0.15  0.22  0.25  0.28  0.35 25253    1
## a[141]     0.13       0 0.05  0.03  0.10  0.13  0.16  0.23 25860    1
## a[142]    -0.10       0 0.07 -0.23 -0.14 -0.09 -0.05  0.05 31393    1
## a[143]     0.00       0 0.09 -0.18 -0.06  0.00  0.05  0.17 27155    1
## a[144]     0.04       0 0.10 -0.15 -0.03  0.04  0.10  0.23 31866    1
## a[145]    -0.05       0 0.07 -0.19 -0.10 -0.05  0.00  0.10 30768    1
## a[146]     0.05       0 0.07 -0.09  0.00  0.05  0.09  0.18 32584    1
## a[147]    -0.03       0 0.09 -0.20 -0.09 -0.03  0.03  0.15 32104    1
## a[148]    -0.06       0 0.07 -0.20 -0.11 -0.06 -0.02  0.07 23844    1
## a[149]     0.05       0 0.06 -0.07  0.01  0.05  0.09  0.16 25550    1
## a[150]    -0.10       0 0.05 -0.21 -0.14 -0.10 -0.07  0.00 26842    1
## a[151]    -0.03       0 0.07 -0.17 -0.08 -0.03  0.02  0.11 20689    1
## a[152]     0.04       0 0.06 -0.07  0.01  0.04  0.08  0.16 25432    1
## a[153]     0.03       0 0.08 -0.13 -0.03  0.03  0.08  0.18 27646    1
## a[154]    -0.04       0 0.08 -0.20 -0.09 -0.04  0.02  0.12 28401    1
## a[155]     0.22       0 0.06  0.11  0.18  0.22  0.26  0.34 27742    1
## a[156]    -0.11       0 0.07 -0.25 -0.16 -0.11 -0.06  0.03 29428    1
## a[157]     0.14       0 0.06  0.01  0.10  0.14  0.19  0.27 27153    1
## a[158]     0.19       0 0.07  0.06  0.14  0.18  0.23  0.31 28310    1
## a[159]     0.05       0 0.07 -0.09  0.00  0.05  0.09  0.19 26350    1
## a[160]    -0.02       0 0.10 -0.21 -0.08 -0.02  0.05  0.17 27033    1
## a[161]    -0.01       0 0.10 -0.20 -0.08 -0.01  0.06  0.18 28338    1
## a[162]    -0.11       0 0.09 -0.28 -0.16 -0.11 -0.05  0.06 29768    1
## a[163]     0.02       0 0.08 -0.14 -0.04  0.02  0.07  0.18 28463    1
## a[164]     0.02       0 0.08 -0.14 -0.04  0.02  0.08  0.18 27788    1
##
## Samples were drawn using NUTS(diag_e) at Mon Dec 06 20:33:16 2021.
## 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).
```

As we can see $\widehat{R}$ are all < 1.05 indicating the chains have converged. Considering the length of the dataset $n = 8552$, also ESS (n_eff) look good for every estimate. Divergent transitions and tree depths for each model can be seen here:

```
#separate
summary(do.call(rbind, get_sampler_params(fit_separate, inc_warmup = FALSE)))
```

```
##  accept_stat__      stepsize__      treedepth__  n_leapfrog__  divergent__
##  Min.   :0.2470   Min.   :0.5078   Min.   :3    Min.   :7     Min.   :0
##  1st Qu.:0.7606   1st Qu.:0.5177   1st Qu.:3    1st Qu.:7     1st Qu.:0
##  Median :0.8823   Median :0.5209   Median :3    Median :7     Median :0
##  Mean   :0.8491   Mean   :0.5196   Mean   :3    Mean   :7     Mean   :0
##  3rd Qu.:0.9657   3rd Qu.:0.5229   3rd Qu.:3    3rd Qu.:7     3rd Qu.:0
##  Max.   :1.0000   Max.   :0.5289   Max.   :3    Max.   :7     Max.   :0
##     energy__
```

```
##  Min.   :243.0
##  1st Qu.:259.8
##  Median :264.6
##  Mean   :264.9
##  3rd Qu.:269.7
##  Max.   :294.3
```

```
#hierarchical
summary(do.call(rbind, get_sampler_params(fit_hierarchical, inc_warmup = FALSE)))
```

```
##  accept_stat__         stepsize__      treedepth__     n_leapfrog__
##  Min.   :0.0000056   Min.   :0.2896   Min.   :3.000   Min.   : 7.00
##  1st Qu.:0.8405993   1st Qu.:0.3274   1st Qu.:4.000   1st Qu.:15.00
##  Median :0.9270193   Median :0.3444   Median :4.000   Median :15.00
##  Mean   :0.8830333   Mean   :0.3397   Mean   :3.933   Mean   :14.71
##  3rd Qu.:0.9738765   3rd Qu.:0.3567   3rd Qu.:4.000   3rd Qu.:15.00
##  Max.   :1.0000000   Max.   :0.3804   Max.   :4.000   Max.   :31.00
##   divergent__    energy__
##  Min.   :0    Min.   :125.3
##  1st Qu.:0    1st Qu.:151.3
##  Median :0    Median :156.6
##  Mean   :0    Mean   :156.8
##  3rd Qu.:0    3rd Qu.:162.5
##  Max.   :0    Max.   :188.6
```

```
#pooled
summary(do.call(rbind, get_sampler_params(fit_pooled, inc_warmup = FALSE)))
```

```
##  accept_stat__      stepsize__      treedepth__     n_leapfrog__    divergent__
##  Min.   :0.2061   Min.   :0.7700   Min.   :1.000   Min.   :1.000   Min.   :0
##  1st Qu.:0.8533   1st Qu.:0.8319   1st Qu.:2.000   1st Qu.:3.000   1st Qu.:0
##  Median :0.9472   Median :0.8961   Median :2.000   Median :3.000   Median :0
##  Mean   :0.9034   Mean   :0.8757   Mean   :1.798   Mean   :3.236   Mean   :0
##  3rd Qu.:0.9957   3rd Qu.:0.9398   3rd Qu.:2.000   3rd Qu.:3.000   3rd Qu.:0
##  Max.   :1.0000   Max.   :0.9405   Max.   :3.000   Max.   :7.000   Max.   :0
##     energy__
##  Min.   :328.0
##  1st Qu.:328.9
##  Median :329.6
##  Mean   :329.9
##  3rd Qu.:330.6
##  Max.   :337.8
```

```
#separate with additional parameter
summary(do.call(rbind, get_sampler_params(fit_separate_id, inc_warmup = FALSE)))
```

```
##  accept_stat__      stepsize__      treedepth__  n_leapfrog__  divergent__
##  Min.   :0.3473   Min.   :0.3422   Min.   :3    Min.   :15    Min.   :0
##  1st Qu.:0.7756   1st Qu.:0.3454   1st Qu.:4    1st Qu.:15    1st Qu.:0
##  Median :0.8863   Median :0.3495   Median :4    Median :15    Median :0
##  Mean   :0.8571   Mean   :0.3536   Mean   :4    Mean   :15    Mean   :0
##  3rd Qu.:0.9637   3rd Qu.:0.3577   3rd Qu.:4    3rd Qu.:15    3rd Qu.:0
```

```
##  Max.   :1.0000   Max.   :0.3732   Max.   :4    Max.   :31    Max.   :0
##     energy__
##  Min.   :179.6
##  1st Qu.:220.0
##  Median :229.8
##  Mean   :230.3
##  3rd Qu.:239.9
##  Max.   :290.6
```

As we can see, there are no divergent transitions and treep depth is reasonable since it's maximum value is easily under 10 for each model.

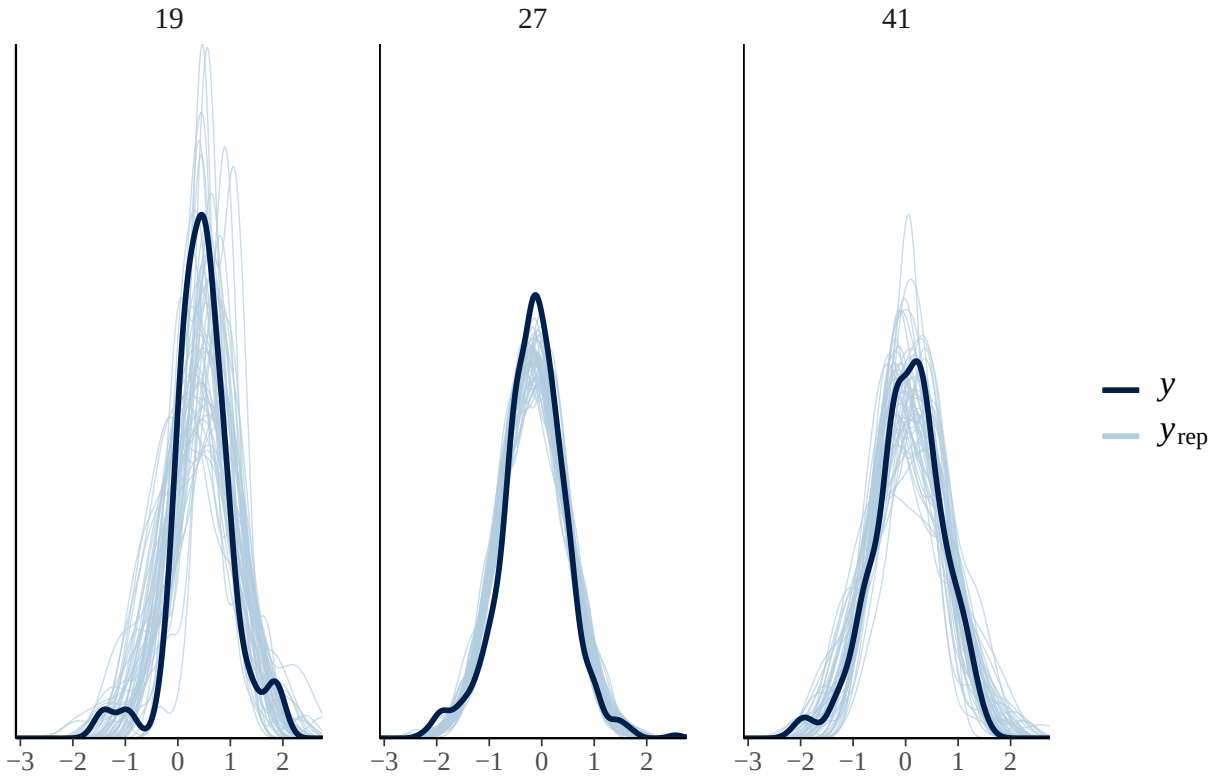
Initially stan warned about few divergent transitions and low ESS for separate model with additional parameter, so we increased the chaing length to 6000 which fixed the problem. We also attempted to fit the hierarchical model with additional parameter but it had several problems with fitting including divergent transitions, big $\widehat{R}$ and low ESS which we were not able to fix by increasing chain length and increasing the adapt_delta parameter for stan, indicating that it's not possible to fit the model at least with our computational resources and within a reasonable amount of time.

Posterior Predictive Checks

Here we plot density plots for replicated dataset y_{rep} together with the observed data y for three different ages, 19, 27 and 41. The age pools 19 and 41 are very sparse, containting only 48 and 57 samples, respectively, while age 27 contains 594 samples.

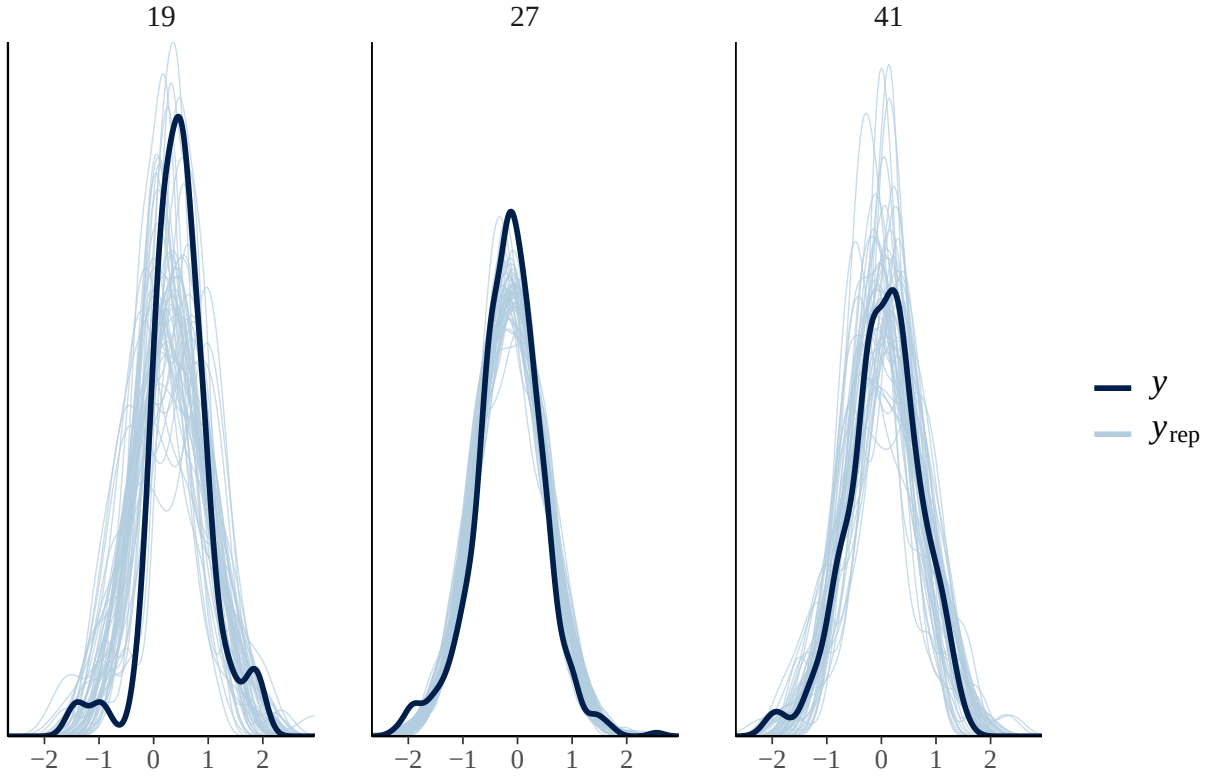
```
yrep_separate <- extract(fit_separate)$yrep
yrep_hierarchical <- extract(fit_hierarchical)$yrep
yrep_pooled <- extract(fit_pooled)$yrep
yrep_separate_id <- extract(fit_separate_id, pars='yrep')$yrep
yrep_separate_id_known <- extract(fit_separate_id, pars='yrep_id')$yrep_id

ppc_dens_overlay_grouped(y, yrep_separate[1:50,], y_rep_groups) + ggtitle('Separate model')
```

Separate model



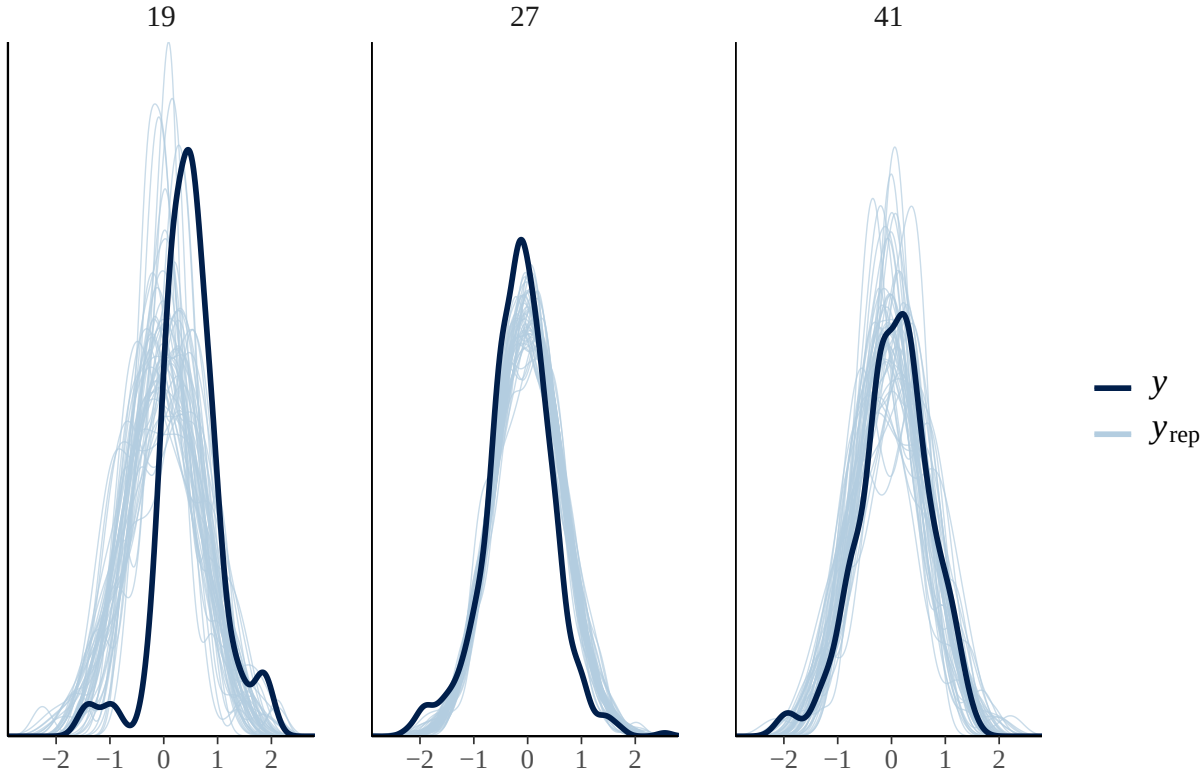

```
ppc_dens_overlay_grouped(y, yrep_hierarchical[1:50,], y_rep_groups) + ggtitle('Hierarchical model')
```

Hierarchical model

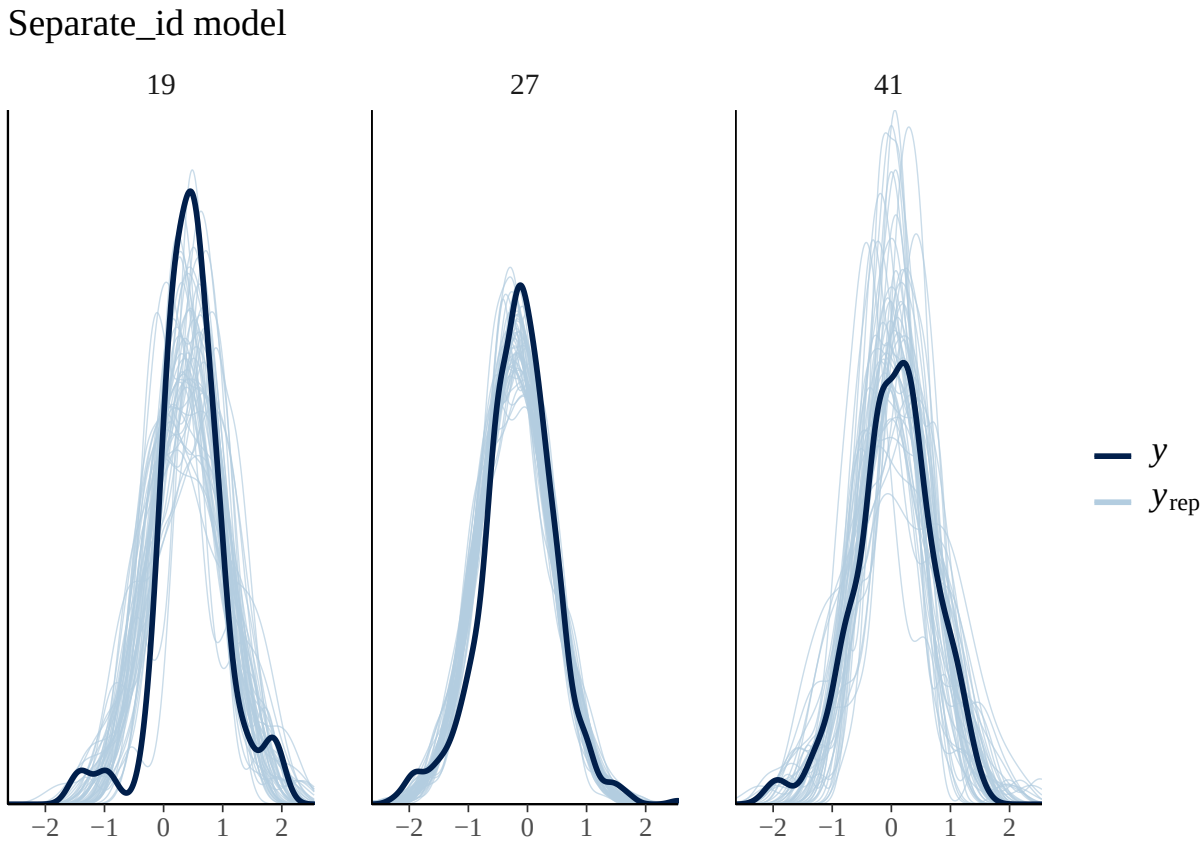



```
ppc_dens_overlay_grouped(y, yrep_pooled[1:50,], y_rep_groups) + ggtitle('Pooled model')
```

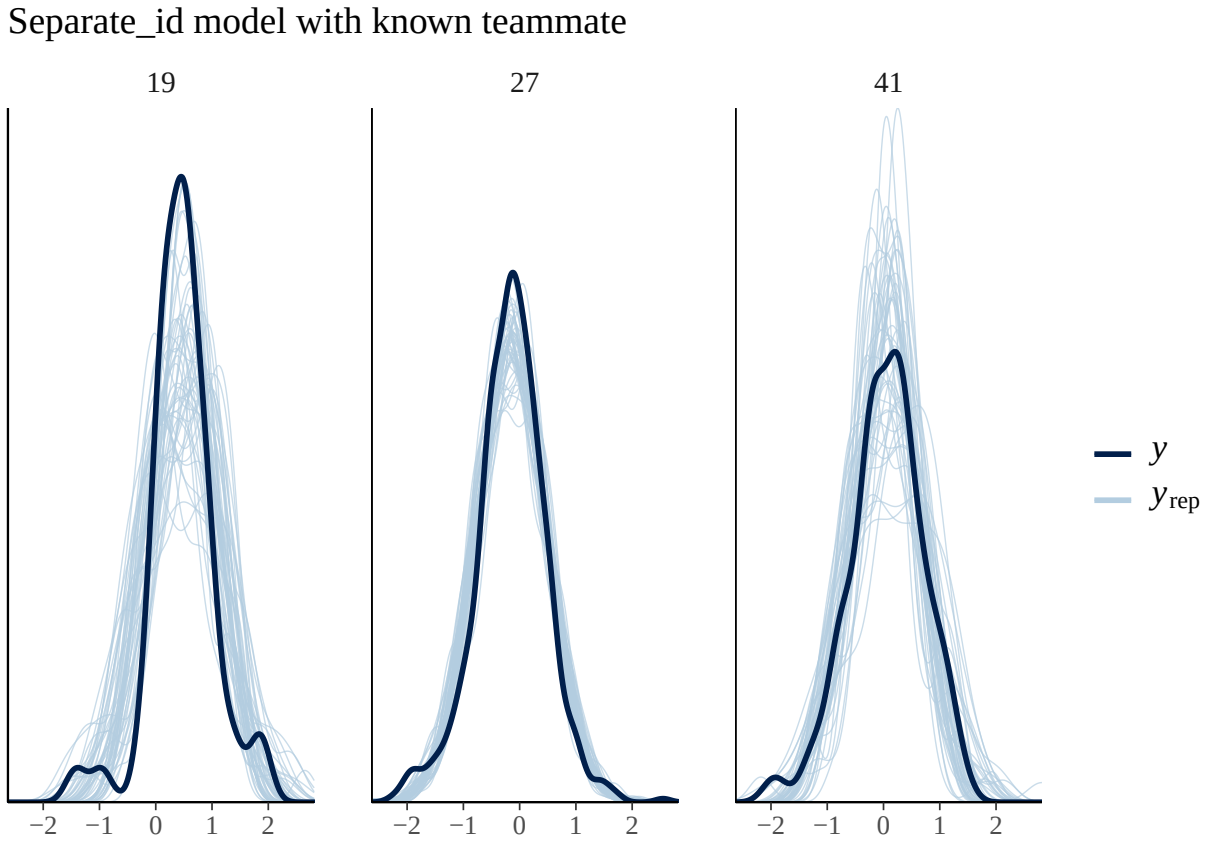
Pooled model




```
ppc_dens_overlay_grouped(y, yrep_separate_id[1:50,], y_rep_groups) + ggtitle('Separate_id model')
```




```
ppc_dens_overlay_grouped(y, yrep_separate_id_known[1:50,], y_rep_groups) + ggtitle('Separate_id model w
```



Based on these plots, hierarchical and separate model seem to have fit reasonably well to the data, while pooled seems to have pretty clear problems especially in age 19. This is expected since we expect age to have some effect to average results. Though, even if it didn't have any effect, it's still possible that the different age pools have differing distributions just by chance, especially as some of them are very sparse, in which case the hierarchical and separate model would fit better anyway. This would be a case of overfit if the reality is that age does not have any effect.

Model Comparison

```
loo_mat_separate <- loo(extract_log_lik(fit_separate))
```

```
## Warning: Relative effective sample sizes ('r_eff' argument) not specified.
## For models fit with MCMC, the reported PSIS effective sample sizes and
## MCSE estimates will be over-optimistic.
```

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

```
loo_mat_hierarchical <- loo(extract_log_lik(fit_hierarchical))
```

```
## Warning: Relative effective sample sizes ('r_eff' argument) not specified.
## For models fit with MCMC, the reported PSIS effective sample sizes and
## MCSE estimates will be over-optimistic.
```

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

```
loo_mat_pooled <- loo(extract_log_lik(fit_pooled))
```

```
## Warning: Relative effective sample sizes ('r_eff' argument) not specified.
## For models fit with MCMC, the reported PSIS effective sample sizes and
## MCSE estimates will be over-optimistic.
```

```
loo_mat_separate_id <- loo(extract_log_lik(fit_separate_id))
```

```
## Warning: Relative effective sample sizes ('r_eff' argument) not specified.
## For models fit with MCMC, the reported PSIS effective sample sizes and
## MCSE estimates will be over-optimistic.

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

```
loo_mat_separate_id_known <- loo(extract(fit_separate_id, pars='log_lik_id')$log_lik_id)
```

```
## Warning: Relative effective sample sizes ('r_eff' argument) not specified.
## For models fit with MCMC, the reported PSIS effective sample sizes and
## MCSE estimates will be over-optimistic.

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

```
loo_compare(loo_mat_hierarchical, loo_mat_pooled, loo_mat_separate, loo_mat_separate_id, loo_mat_separa
```

```
##        elpd_diff se_diff
## model5    0.0       0.0
## model1 -195.4      17.4
## model3 -200.4      16.7
## model2 -274.0      23.5
## model4 -286.4      20.1
```

Predictive Performance Assessment

...

Prior Sensitivity Analysis

...

Discussion

...

Conclusions and Self-Reflections

...

References

"Ergast Developer API." n.d. Accessed November 30, 2021. http://ergast.com/mrd/.

Jain, Adit. n.d. "Predicting the Results for the World's Most Expensive Sport." Accessed December 1, 2021. https://medium.com/@ajleo899/predicting-the-results-for-the-worlds-most-expensive-sport-102e7bc97b25.

Nigro, Veronica. n.d. "Formula 1 Race Predictor." Accessed November 30, 2021. https://towardsdatascience.com/formula-1-race-predictor-5d4bfae887da.

van Kesteren, Erik-Jan. n.d. "Bayesian Analysis of Formula One Race Results." Accessed December 1, 2021. https://github.com/vankesteren/f1model.

Vehtari, Aki, Andrew Gelman, Daniel Simpson, Bob Carpenter, and Paul-Christian Bürkner. 2021. "Rank-Normalization, Folding, and Localization: An Improved r^ for Assessing Convergence of MCMC (with Discussion)." *Bayesian Analysis* 16 (2). https://doi.org/10.1214/20-ba1221.