BDA Project

Anonymous

Contents

1 Introduction

2 Data

The data used in this project comes from the "Advancing Sustainable Materials Management: Facts and Figures Report" of the United States Environmental Protection Agency (EPA (2022)). The data is available for download through this link: https://edg.epa.gov/data/PUBLIC/OLEM/Materials_Municipal_Waste_Stream_1960_2018.xlsx.

The data is in Excel format and contains waste management data from the US for the years 1960-2018. This document contains a large amount of sheets, but for this project, only the "Materials generated", "Materials recycled", "Material combusted", and "Materials landfilled" sheets were used, as they are the only ones with clear data for textiles.

The data from 1960 to 2000 is given every ten years, then there is data for 2005, and the data from 2010-2018 is given yearly. Due to this inconsistency in time intervals of data, only part of the years are used to fit the model. The years 2010-2018 were chosen, as they have the most regular data and they are the most recent. The rest of the data is used to help in estimating prior parameters.

Also, the sheet "Materials generated" is not used in the model as it isn't interesting to the goal of the project, but it is used to aid in prior parameter determination.

2.1 The final data

As the data is scattered in multiple sheets of an Excel file, the specific textile-related data had to be extracted from each sheet, combined, and the row and column names cleaned up a bit in order to acquire a final dataset. This results in a dataframe with categories "Materials generated", "Materials recycled", "Material combusted".

The year is also adjusted such that the year 1950 becomes the new year 0 (1950 is subtracted from each year). This is done so that the significantly long period of time before data collection where values for each category might have been 0 doesn't affect the intercept or the slope of the final models. For example, the first landfill in the US was established in 1937 (Zylberberg (2019)), so there would have been 1937 years of 0 values in the "Materials landfilled" catefgory, which could have a significant impact on the linear model. Hence, a new year 0 was chosen in which all categories of waste management had already been implemented in some sense.

The final dataframe can be seen in Table 1.

The final data is visualised in Figure 1 for years 1960-2005, and in Figure 2 for years 2010-2018.

As can be seen from the figures, the data could likely be approximated decently well with a linear model.

Table 1: Final dataframe

Materials generated	Materials recycled	Material combusted	Materials landfilled	year	original_year
1760000	50000	0	1710000	10	1960
2040000	60000	10000	1970000	20	1970
2530000	160000	50000	2320000	30	1980
5810000	660000	880000	4270000	40	1990
9480000	1320000	1880000	6280000	50	2000
11510000	1830000	2110000	7570000	55	2005
13220000	2050000	2270000	8900000	60	2010
13690000	2070000	2540000	9080000	61	2011
14480000	2200000	2800000	9480000	62	2012
14840000	2220000	2960000	9660000	63	2013
15240000	2260000	3020000	9960000	64	2014
16060000	2460000	3060000	10540000	65	2015
16880000	2510000	3240000	11130000	66	2016
16890000	2570000	3170000	11150000	67	2017
17030000	2510000	3220000	11300000	68	2018

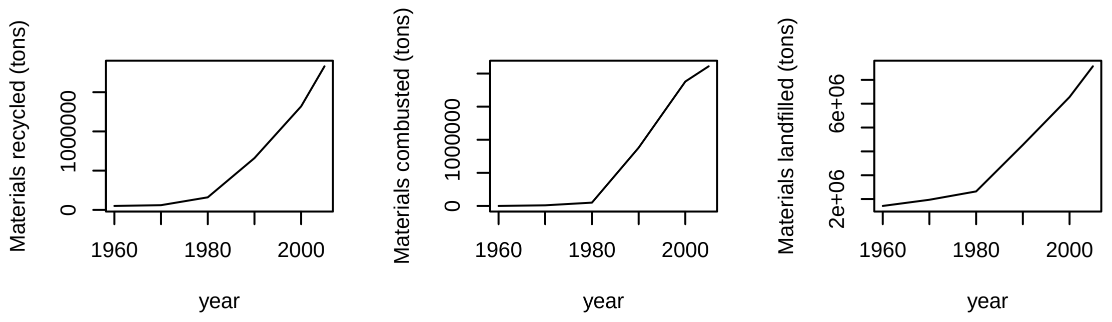


Figure 1: Data.

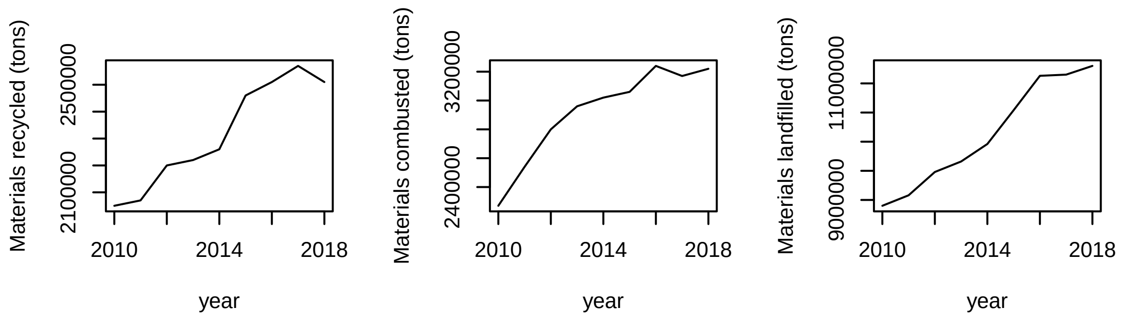


Figure 2: Data.

3 The models

Since the data used in this project is split into three categories (materials recycled, materials combusted, and materials landfilled), and their differences are an aspect of interest, a model was needed which differentiates between them. Two good options for this are the separate model and the hierarchical model. Hence, in this project both of these models are implemented and compared.

3.1 Separate model

The separate model assumes that each category is independent from the other categories, and essentially gives each category its own Bayesian model. Hence, it considers the categories to have no relation between each other,

3.2 Hierarchical model

The hierarchical model considers each category as being distinct from each other, but still similar enough that the measurements from one category can influence the predictions for another. To accommodate for this, the hierarchical model provides separate models for each category, however the models share the same variance and the means of the models share the same prior distribution constructed with hyperparameters. Hence, the hierarchical model considers the categories to be distinct, but still aknowledges some relation between them.

3.3 Linear Gaussian model

As was shown in the "Data" section, the data used to fit the models is linear, with the explanatory variable being the year. Hence, both the separate and hierarchical models are implemented as linear gaussian models in this project. This means the parameters of interest are the intercepts and slopes of the lines describing the data.

4 Priors

4.1 Initial priors

4.2 Final priors

4.3 Parameters

```r
# Separate model:
#Estimate intercept means:
#recycled_intercept <- 680000*0.001 #US population in year 0  times 1kg (0.001 tonnes) for clothing was
#combusted_intercept <- 0
#landfilled_intercept <- 0

#Estimate slope means:
summary_2 <- summary(lm(`Materials recycled` ~ year, data=all_data[(1:6),]))
recycled_slope <- summary_2$coefficients[2,1]

summary_3 <- summary(lm(`Material combusted` ~ year, data=all_data[(1:6),]))
```

```r
combusted_slope <- summary_3$coefficients[2,1]

summary_4 <- summary(lm(`Materials landfilled` ~ year, data=all_data[(1:6),]))
landfilled_slope <- summary_4$coefficients[2,1]

beta_mean = c(recycled_slope, combusted_slope, landfilled_slope)

sd_mean = c(sd(all_data[1:6,'Materials recycled']), sd(all_data[1:6,'Material combusted']), sd(all_data
# Calculate the factor by how much the standard deviation of generated textiles has shrunk after betwee
factor <- sd(all_data[1:6, 'Materials generated'])/sd(all_data[7:15, 'Materials generated'])
# Divide sd's by the factor to make them more realistic
sd_mean <- sd_mean/factor

# Intercept means for 1950 0:
recycled_intercept <- max(summary_2$coefficients[1,1], 0)
combusted_intercept <- max(summary_3$coefficients[1,1], 0)
landfilled_intercept <- max(summary_4$coefficients[1,1], 0)
alpha_mean = c(recycled_intercept, combusted_intercept, landfilled_intercept)

# Hierarchical model:
hierarchical_intercept_mean <- mean(alpha_mean)
hierarchical_slope_mean <- mean(beta_mean)
hierarchical_sd_mean <- mean(sd_mean)
```

5 Stan code

5.1 Separate

5.2 Hierarchical

6 Fitting the models

```r
# Prepare data
data_hierarchical = list(
  N = nrow(all_data[7:15,]),
  J = ncol(all_data[,2:4]),
  x = all_data[7:15,'year'],
  y = all_data[7:15, 2:4],
  xpred = (2019-1950),
  sd_mean = hierarchical_sd_mean,
  slope_mean = hierarchical_slope_mean,
  intercept_mean = hierarchical_intercept_mean
)

data_separate = list(
  N = nrow(all_data[7:15,]),
  J = ncol(all_data[,2:4]),
  x = all_data[7:15,'year'],
  y = all_data[7:15, 2:4],
  xpred = (2019-1950),
  alpha_mean = alpha_mean,
  beta_mean = beta_mean,
  sds = sd_mean
)


fit_hierarchical <- stan(file = "Hierarchical_project.stan", data = data_hierarchical) #Had to remove t
fit_separate <- stan(file = "Separate_project.stan", data = data_separate)
monitor(fit_hierarchical)
monitor(fit_separate)
```

7 Convergence diagnostics

7.1 R-hat and ESS

Hierarchical:

```
hier_matr = matrix(data=NA, nrow=6, ncol=3)
colnames(hier_matr) = c('Rhat', 'bulk-ESS', 'tail-ESS')
rownames(hier_matr) = c('alpha[1]', 'beta[1]', 'alpha[2]', 'beta[2]', 'alpha[3]', 'beta[3]')
for (i in 1:3) {
  alpha <- paste('alpha[', as.character(i), ']', sep="")
  beta <- paste('beta[', as.character(i), ']', sep="")
  alpha_mat <- extract_variable_matrix(fit_hierarchical, alpha)
  beta_mat <- extract_variable_matrix(fit_hierarchical, beta)
  rhat_alpha <- posterior::rhat(alpha_mat)
  rhat_beta <- posterior::rhat(beta_mat)
  b_ess_alpha <- posterior::ess_bulk(alpha_mat)
  b_ess_beta <- posterior::ess_bulk(beta_mat)
  t_ess_alpha <- posterior::ess_tail(alpha_mat)
  t_ess_beta <- posterior::ess_tail(beta_mat)
  hier_matr[i+(i-1), 1] <- rhat_alpha
  hier_matr[i+(i-1)+1, 1] <- rhat_beta
  hier_matr[i+(i-1), 2] <- b_ess_alpha
  hier_matr[i+(i-1)+1, 2] <- b_ess_beta
  hier_matr[i+(i-1), 3] <- t_ess_alpha
  hier_matr[i+(i-1)+1, 3] <- t_ess_beta
}
kable(hier_matr, caption='Hierarchical model')
```

Hierarchical had good Rhat and ESS from the start.

Separate:

```
sep_matr = matrix(data=NA, nrow=6, ncol=3)
colnames(sep_matr) = c('Rhat', 'bulk-ESS', 'tail-ESS')
rownames(sep_matr) = c('alpha[1]', 'beta[1]', 'alpha[2]', 'beta[2]', 'alpha[3]', 'beta[3]')
for (i in 1:3) {
  alpha <- paste('alpha[', as.character(i), ']', sep="")
  beta <- paste('beta[', as.character(i), ']', sep="")
```

Table 2: Hierarchical model

	Rhat	bulk-ESS	tail-ESS
alpha[1]	1.000671	2431.945	1507.388
beta[1]	1.001274	4086.169	2493.827
alpha[2]	1.000136	2597.963	2035.260
beta[2]	0.999734	4172.335	2677.295
alpha[3]	1.000535	2563.618	1684.200
beta[3]	1.000807	4072.720	2336.372

Table 3: Separate model

	Rhat	bulk-ESS	tail-ESS
alpha[1]	1.000991	1856.760	1342.541
beta[1]	1.001449	3340.914	2480.001
alpha[2]	1.002347	2281.318	1373.461
beta[2]	1.001407	3370.391	2577.556
alpha[3]	1.000872	2386.340	1560.344
beta[3]	1.002501	3443.360	2385.659

```
  alpha_mat <- extract_variable_matrix(fit_separate, alpha)
  beta_mat <- extract_variable_matrix(fit_separate, beta)
  rhat_alpha <- posterior::rhat(alpha_mat)
  rhat_beta <- posterior::rhat(beta_mat)
  b_ess_alpha <- posterior::ess_bulk(alpha_mat)
  b_ess_beta <- posterior::ess_bulk(beta_mat)
  t_ess_alpha <- posterior::ess_tail(alpha_mat)
  t_ess_beta <- posterior::ess_tail(beta_mat)
  sep_matr[i+(i-1), 1] <- rhat_alpha
  sep_matr[i+(i-1)+1, 1] <- rhat_beta
  sep_matr[i+(i-1), 2] <- b_ess_alpha
  sep_matr[i+(i-1)+1, 2] <- b_ess_beta
  sep_matr[i+(i-1), 3] <- t_ess_alpha
  sep_matr[i+(i-1)+1, 3] <- t_ess_beta
}
kable(sep_matr, caption='Separate model')
```

For first separate model all Rhat values were >1.05 and most ESS values were <100. Because of this, the model was simplified by removing the sigma prior entirely. This fixed the issue.

7.2 HMC diagnostics

```
check_hmc_diagnostics(fit_hierarchical)
```

```
##
## Divergences:
```

```
## 0 of 4000 iterations ended with a divergence.
```

```
##
## Tree depth:

## 0 of 4000 iterations saturated the maximum tree depth of 10.

##
## Energy:

## E-BFMI indicated no pathological behavior.
```

The hierarchical model had a few divergences in the initial model. Changing adapt_delta didn't seem to do much, so instead priors were modified. At this point, a mistake in the alpha_sigma and beta_sigma priors was noticed. In the separate model, the standard deviation parameters for alpha and beta were too small compared to the values in the data, meaning the priors weren't really weakly informative. The parameters of the alpha_sigma and beta_sigma priors in the hierarchical model are calculated based on these standard deviations in the separate model. This calculation was also nonsensical, additioning all the standard deviations instead of averaging them. Both of these issues were addressed, after which there were no more divergences in the hierarchical model, as can be seen above.

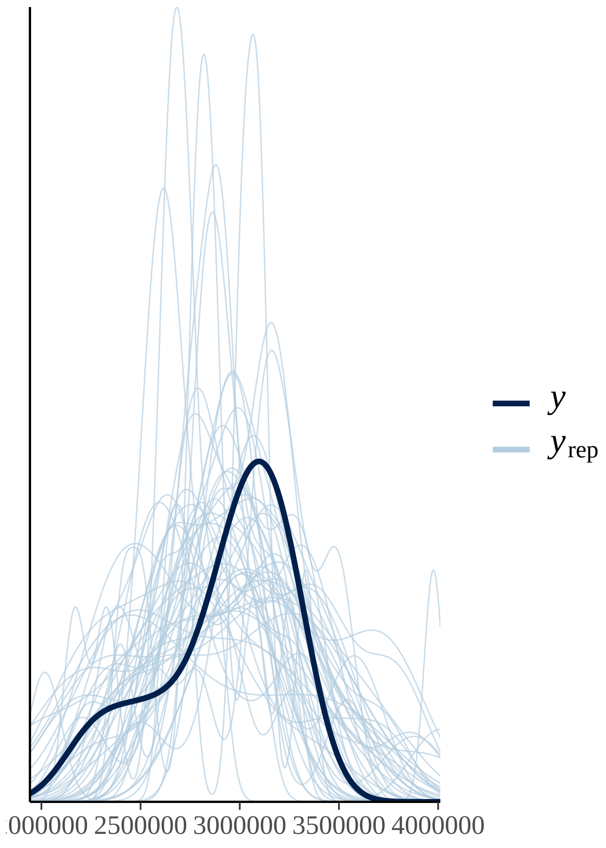
```
check_hmc_diagnostics(fit_separate)
```

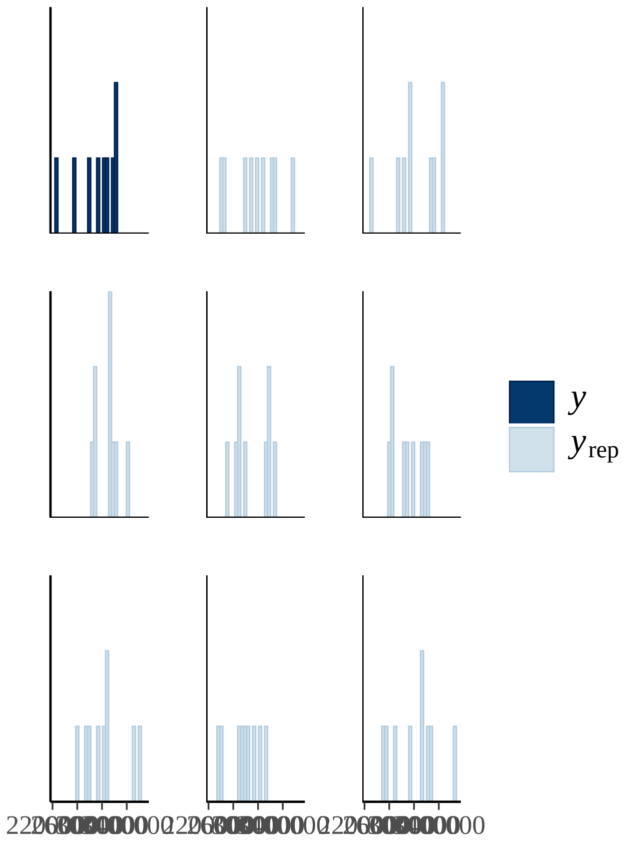
```
##
## Divergences:

## 0 of 4000 iterations ended with a divergence.

##
## Tree depth:

## 0 of 4000 iterations saturated the maximum tree depth of 10.

##
## Energy:

## E-BFMI indicated no pathological behavior.
```

8 Posterior predictive checks

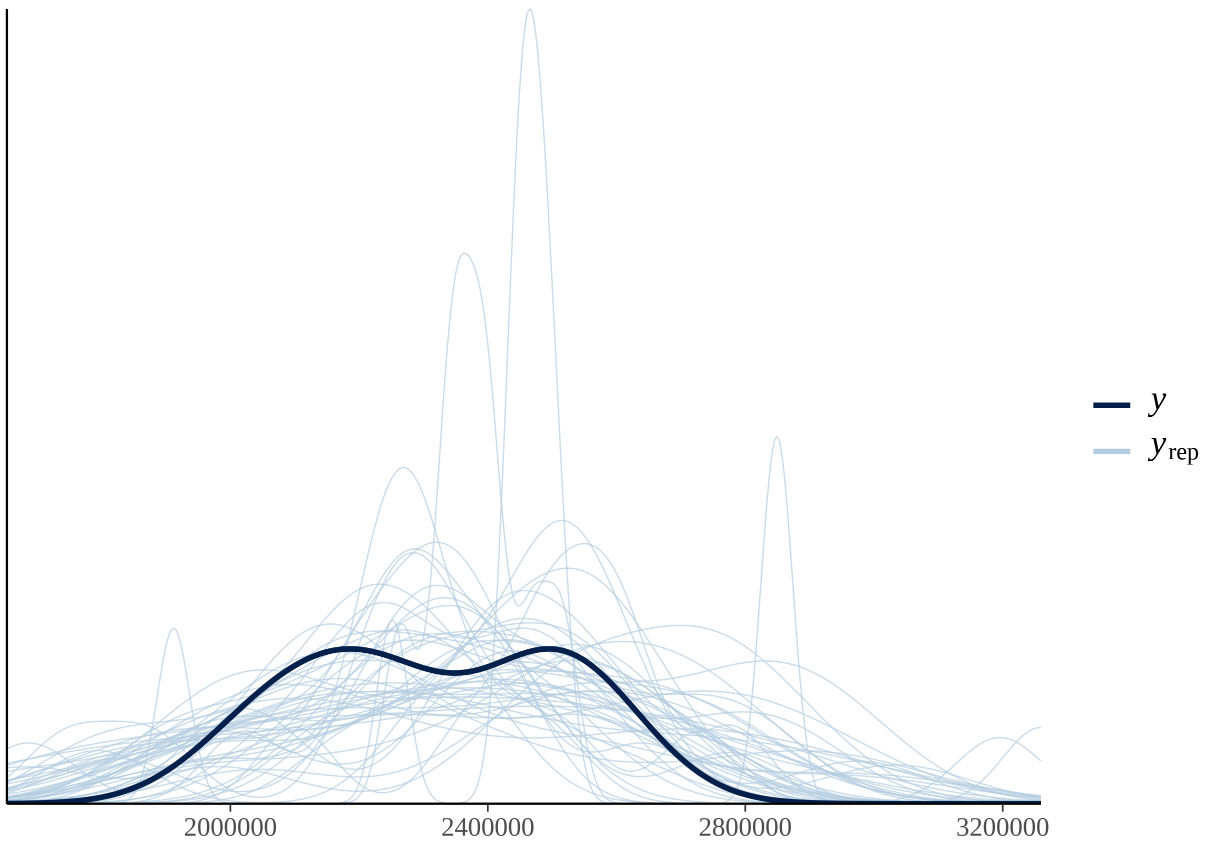
```
# Materials combusted 2019 prediction
s_yrep_c <- as.matrix(fit_separate, pars = "yrep_c")
s_yrep_r <- as.matrix(fit_separate, pars = "yrep_r")
s_yrep_l <- as.matrix(fit_separate, pars = "yrep_l")
h_yrep_c <- as.matrix(fit_hierarchical, pars = "yrep_c")
h_yrep_r <- as.matrix(fit_hierarchical, pars = "yrep_r")
h_yrep_l <- as.matrix(fit_hierarchical, pars = "yrep_l")
y_c <- all_data[['Material combusted']]
y_r <- all_data[['Materials recycled']]
y_l <- all_data[['Materials landfilled']]
par(mfrow=c(1,2))
p_dens <- ppc_dens_overlay(y_c[7:15], yrep = s_yrep_c[1:50,])
p_hist <- ppc_hist(y_c[7:15], yrep = s_yrep_c[1:8,])
plot_grid(p_dens, p_hist)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

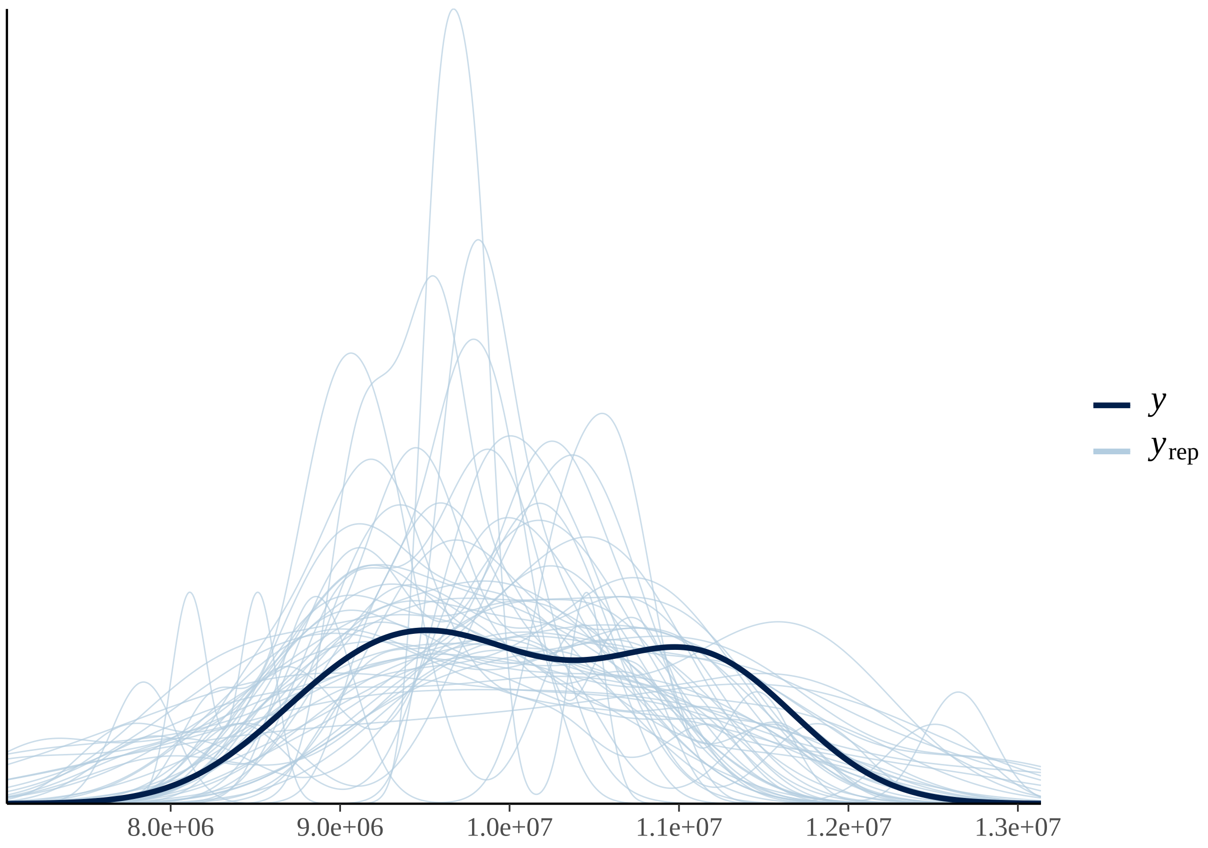




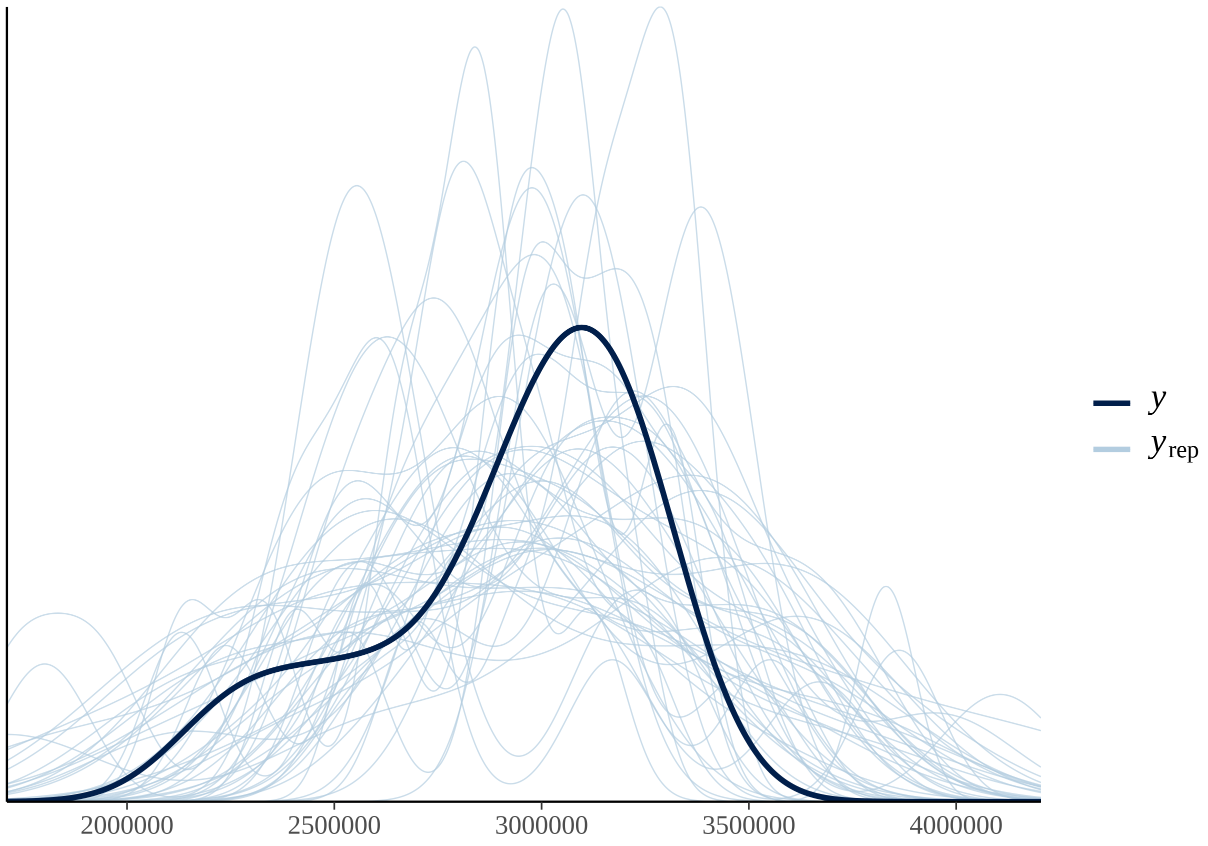
```
ppc_dens_overlay(y_r[7:15], yrep = s_yrep_r[1:50,])
```



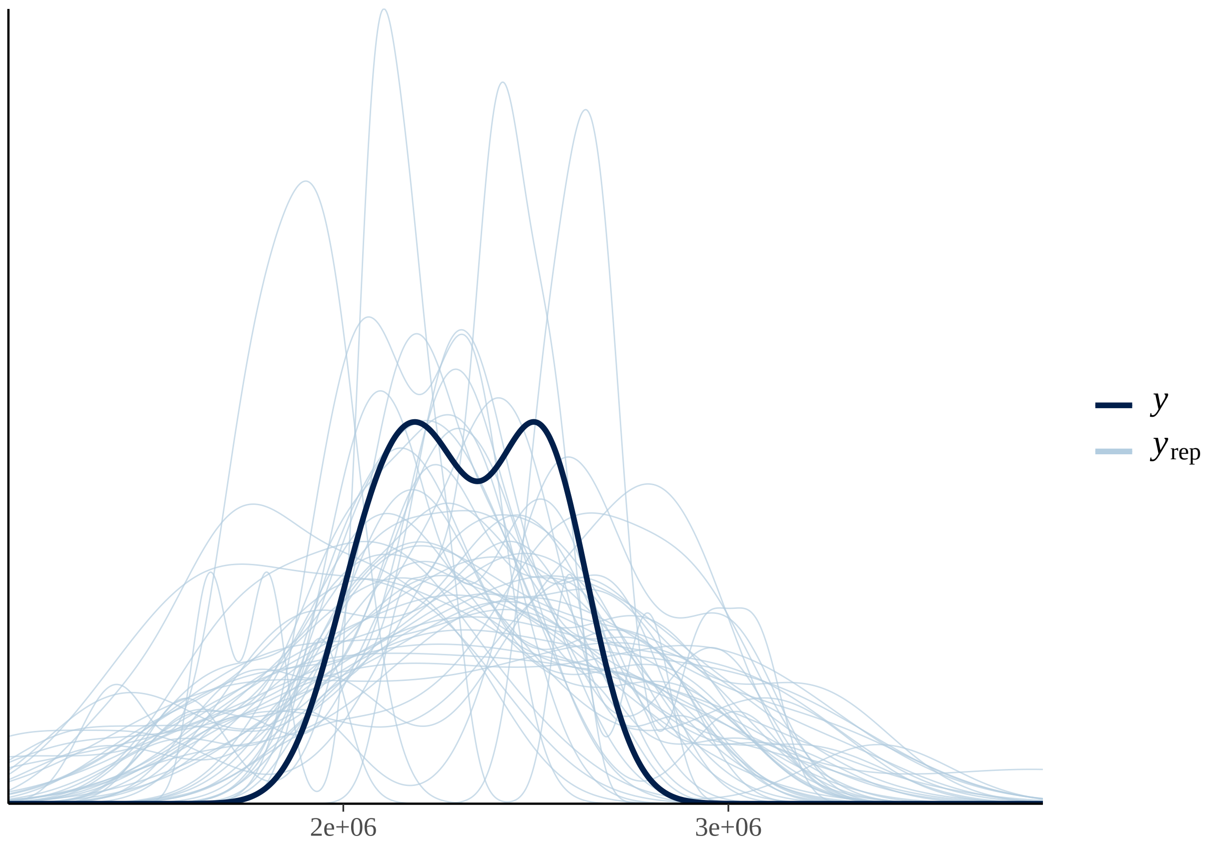
```
ppc_dens_overlay(y_l[7:15], yrep = s_yrep_l[1:50,])
```



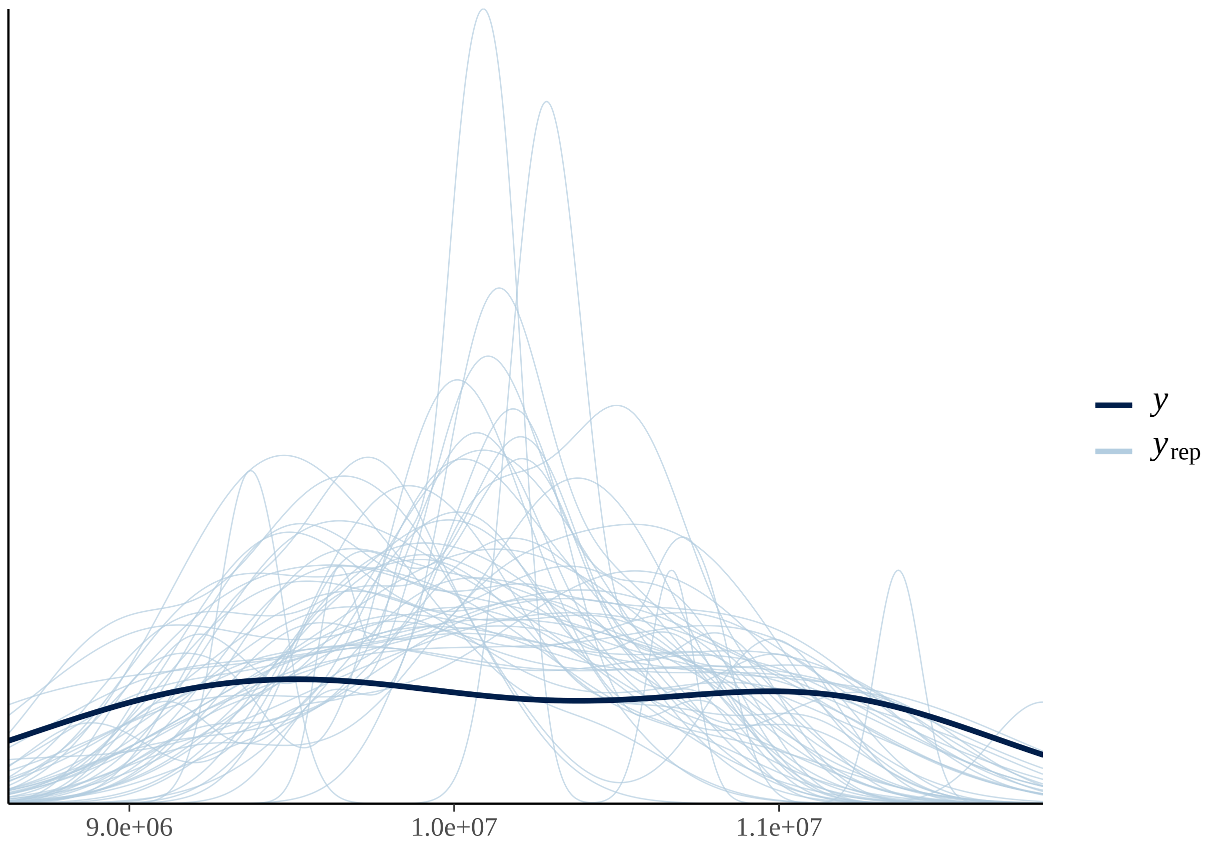
```r
ppc_dens_overlay(y_c[7:15], yrep = h_yrep_c[1:50,])
```



```
ppc_dens_overlay(y_r[7:15], yrep = h_yrep_r[1:50,])
```



```
ppc_dens_overlay(y_l[7:15], yrep = h_yrep_l[1:50,])
```



9 Predictive performance assessment

To assess performance, we can fit the model excluding the year 2018, instead adding that year as the predicted year. We can then compare the true values for 2018 to the predicted values.

Fitting the models:

```
# Prepare data
performance_data_hierarchical = list(
  N = nrow(all_data[7:14,]),
  J = ncol(all_data[,2:4]),
  x = all_data[7:14,'year'],
  y = all_data[7:14, 2:4],
  xpred = (2018-1950),
  sd_mean = hierarchical_sd_mean,
  slope_mean = hierarchical_slope_mean,
  intercept_mean = hierarchical_intercept_mean
)

performance_data_separate = list(
  N = nrow(all_data[7:14,]),
  J = ncol(all_data[,2:4]),
  x = all_data[7:14,'year'],
  y = all_data[7:14, 2:4],
  xpred = (2018-1950),
```

```r
  alpha_mean = alpha_mean,
  beta_mean = beta_mean,
  sds = sd_mean
)


fit_hierarchical_perf <- stan(file = "Hierarchical_project.stan", data = performance_data_hierarchical)
```

```
##
## SAMPLING FOR MODEL 'Hierarchical_project' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 2.1e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.21 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.287871 seconds (Warm-up)
## Chain 1:                0.093767 seconds (Sampling)
## Chain 1:                0.381638 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'Hierarchical_project' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 9e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.09 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.227479 seconds (Warm-up)
## Chain 2:                0.084268 seconds (Sampling)
## Chain 2:                0.311747 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'Hierarchical_project' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 1.1e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.11 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.240767 seconds (Warm-up)
## Chain 3:                0.092577 seconds (Sampling)
## Chain 3:                0.333344 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'Hierarchical_project' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 1.3e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.13 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.296928 seconds (Warm-up)
## Chain 4:                0.104865 seconds (Sampling)
## Chain 4:                0.401793 seconds (Total)
## Chain 4:
```

```
fit_separate_perf <- stan(file = "Separate_project.stan", data = performance_data_separate)
```

```
##
## SAMPLING FOR MODEL 'Separate_project' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 2.2e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.22 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.362853 seconds (Warm-up)
## Chain 1:                0.077937 seconds (Sampling)
## Chain 1:                0.44079 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'Separate_project' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 8e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.08 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.3433 seconds (Warm-up)
## Chain 2:                0.088753 seconds (Sampling)
## Chain 2:                0.432053 seconds (Total)
## Chain 2:
```

```
##
## SAMPLING FOR MODEL 'Separate_project' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 8e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.08 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.351388 seconds (Warm-up)
## Chain 3:                0.059937 seconds (Sampling)
## Chain 3:                0.411325 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'Separate_project' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 8e-06 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.08 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.344012 seconds (Warm-up)
## Chain 4:                0.077599 seconds (Sampling)
## Chain 4:                0.421611 seconds (Total)
## Chain 4:
```

```
## Warning: 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.
```

Table 4: Hierarchical model

	Materials recycled	Materials combusted	Materials landfilled
Real	2510000	3220000.0	11300000.0
Predicted mean	2496674	3130185.0	10620435.4
Predicted sd	414029	403973.3	415921.3

Table 5: Separate model

	Materials recycled	Materials combusted	Materials landfilled
Real	2510000.0	3220000.0	11300000
Predicted mean	2462968.2	3115447.9	10429106
Predicted sd	269553.5	348863.1	907132

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

Results:

```
ypred_r_h <- extract(fit_hierarchical_perf, 'ypred_r')
ypred_c_h <- extract(fit_hierarchical_perf, 'ypred_c')
ypred_l_h <- extract(fit_hierarchical_perf, 'ypred_l')
pred_h <- c(ypred_r_h, ypred_c_h, ypred_l_h)
ypred_r_s <- extract(fit_separate_perf, 'ypred_r')
ypred_c_s <- extract(fit_separate_perf, 'ypred_c')
ypred_l_s <- extract(fit_separate_perf, 'ypred_l')
pred_s <- c(ypred_r_s, ypred_c_s, ypred_l_s)

real_vals <- all_data[all_data$year==(2018 - 1950),]
reals <- c(real_vals[['Materials recycled']], real_vals[['Material combusted']], real_vals[['Materials

hier_pred = matrix(data=NA, nrow=3, ncol=3)
colnames(hier_pred) = c('Materials recycled', 'Materials combusted', 'Materials landfilled')
rownames(hier_pred) = c('Real', 'Predicted mean', 'Predicted sd')
for (i in 1:3) {
  hier_pred[1,i] <- reals[[i]]
  hier_pred[2,i] <- mean(pred_h[[i]])
  hier_pred[3,i] <- sd(pred_h[[i]])
}
kable(hier_pred, caption='Hierarchical model')
```

```
sep_pred = matrix(data=NA, nrow=3, ncol=3)
colnames(sep_pred) = c('Materials recycled', 'Materials combusted', 'Materials landfilled')
rownames(sep_pred) = c('Real', 'Predicted mean', 'Predicted sd')
for (i in 1:3) {
  sep_pred[1,i] <- reals[[i]]
  sep_pred[2,i] <- mean(pred_s[[i]])
  sep_pred[3,i] <- sd(pred_s[[i]])
}
kable(sep_pred, caption='Separate model')
```

10 Sensitivity analysis

10.1 Hierarchical

```
ggplot(data=hier_sens_df, aes(x=value, color = Prior)) +
  geom_density() +
  facet_wrap(y_type ~ col_val, scales='free')
```

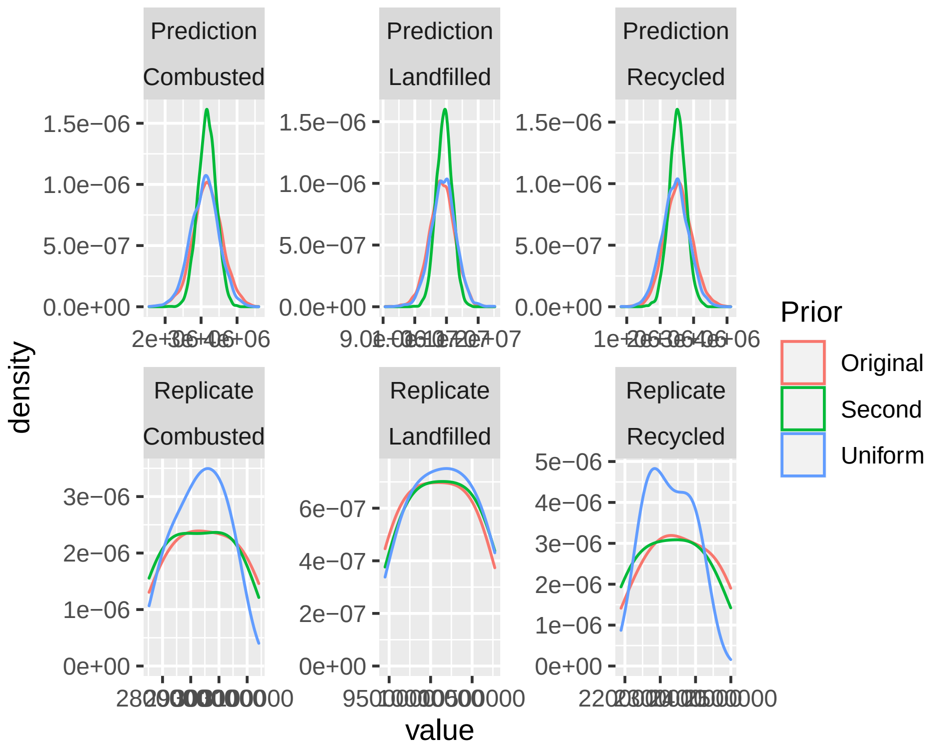


Figure 3: (#fig:hier__sens__plot)Sensitivity analysis for hierarchical model.

10.2 Separate

```
ggplot(data=sep_sens_df, aes(x=value, color = Prior)) +
  geom_density() +
  facet_wrap(y_type ~ col_val, scales='free')
```

11 Model comparison (LOO-CV)

```
#PSIS-LOO for separate model
sep_loo <- rstan::loo(fit_separate)
```

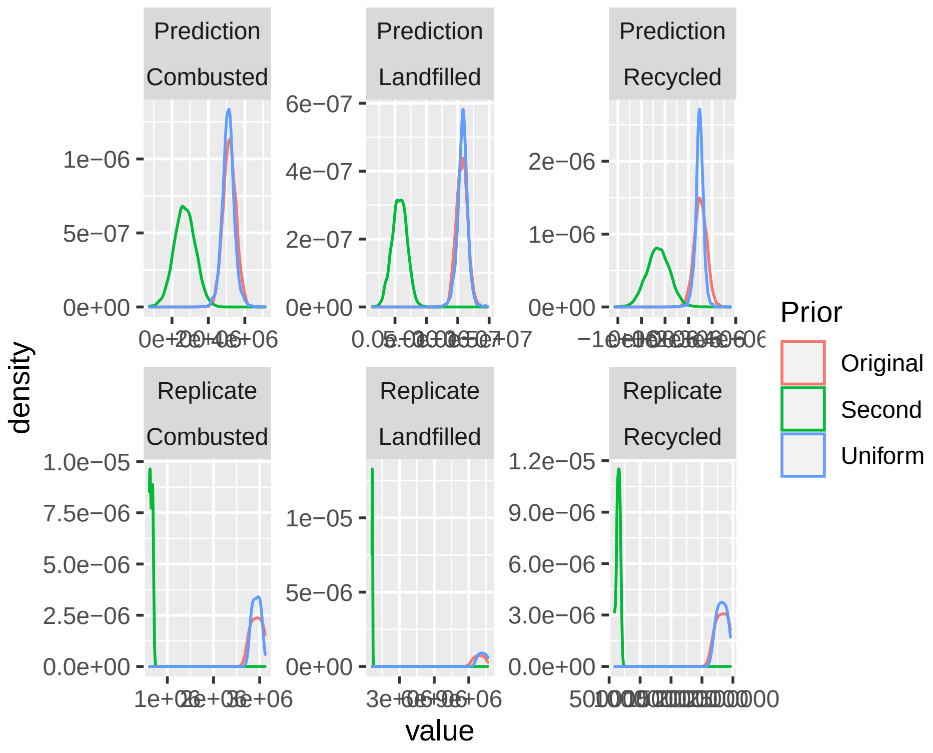


Figure 4: (#fig:sep_sens_plot)Sensitivity analysis for separate model.

```r
sep_elpd <- sep_loo$estimates[1,1]
# Effective number of parameters:
sep_p <- sep_loo$estimates[2,1]
plot(sep_loo)
```

```r
#PSIS-LOO for hierarchical model
hier_loo <- rstan::loo(fit_hierarchical)
```

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

```r
hier_elpd <- hier_loo$estimates[1,1]
# Effective number of parameters:
hier_p <- hier_loo$estimates[2,1]
plot(hier_loo)
```

```r
loo_table <- rbind(c(hier_elpd, sep_elpd), c(hier_p, sep_p))
colnames(loo_table) <- c('Hierarchical', 'Separate')
rownames(loo_table) <- c('elpd', 'p_eff')
kable(loo_table, caption='LOO-CV results')
```

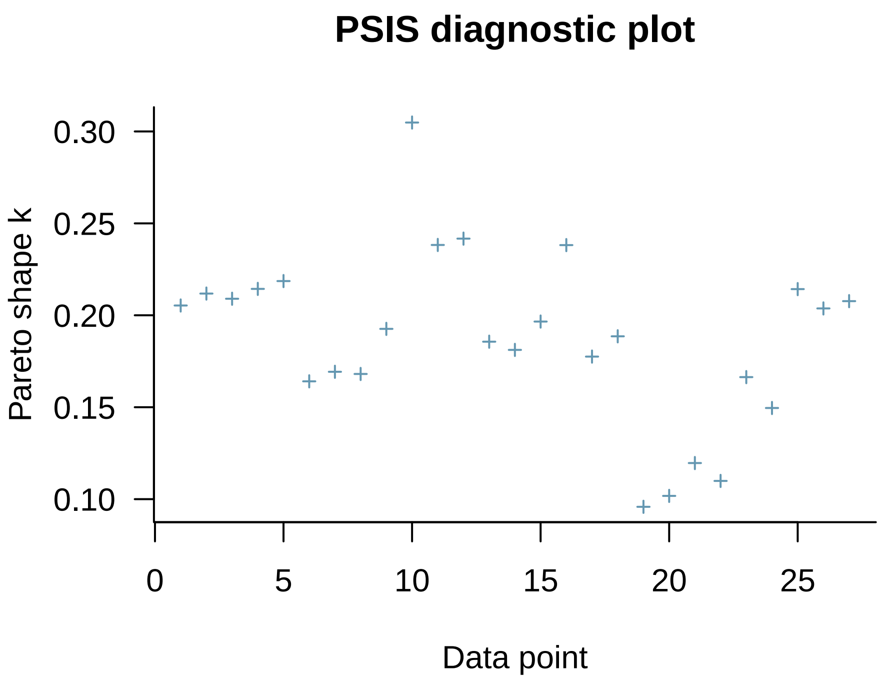


Figure 5: (#fig:separate_k_plot)k-hat values for the separate model.

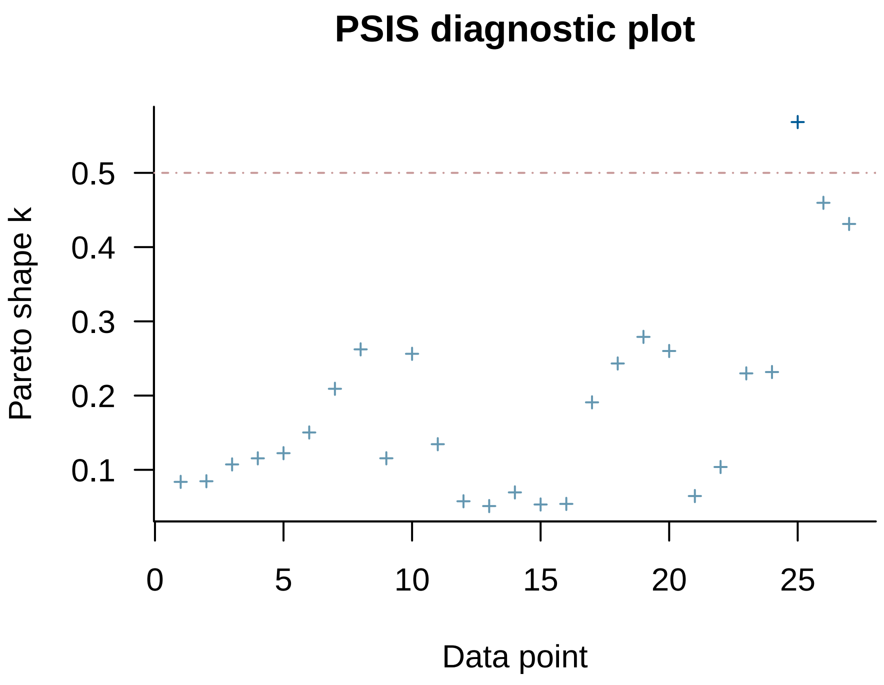


Figure 6: (#fig:hierarchical_k_plot)k-hat values for the hierarchical model.

Table 6: LOO-CV results

	Hierarchical	Separate
elpd	-384.12633	-380.380277
p_eff	3.11968	1.014588

12 Discussion

13 Conclusion

14 Self reflection

References

EPA. 2022. "Advancing Sustainable Materials Management: Facts and Figures Report." *EPA*. Environmental Protection Agency. https://www.epa.gov/facts-and-figures-about-materials-waste-and-recycling/advancing-sustainable-materials-management.

Zylberberg, Nadine. 2019. "The Origin &Amp; Future of Landfill." *Medium*. 2030 magazine-the stuff of stars. https://medium.com/2030magazine/whats-the-future-of-landfills-334370b3a538.