

```
library(rstan)
library(rstanarm)
library(tidyverse)
```

```
m333 <- read_csv("../data/subsets/dat_small.csv") %>%
  filter(province == "M333")
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   subsection = col_character(),
##   section = col_character(),
##   province = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

```
mod <- stan_lmer(BIOLIVE_TPA ~ 1 + nlcd11 + (1 | subsection),
  data = m333,
  verbose = FALSE)
```

```
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.000366 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 3.66 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: 8.3409 seconds (Warm-up)
## Chain 1:                4.35379 seconds (Sampling)
## Chain 1:                12.6947 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0.000142 seconds
```

```

## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 1.42 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: 8.95005 seconds (Warm-up)
## Chain 2:                3.1938 seconds (Sampling)
## Chain 2:                12.1439 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0.000163 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 1.63 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: 7.46587 seconds (Warm-up)
## Chain 3:                2.7801 seconds (Sampling)
## Chain 3:                10.246 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0.000171 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 1.71 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: 8.08106 seconds (Warm-up)
## Chain 4:           3.64154 seconds (Sampling)
## Chain 4:           11.7226 seconds (Total)
## Chain 4:
```

```
mod_df <- data.frame(
  fitted = mod$fitted.values,
  subsection = m333 %>% dplyr::select(subsection),
  true = mod$y
)
```

```
mod_df %>%
  group_by(subsection) %>%
  summarize(bayes_cv = sd(fitted) / mean(fitted),
            direct_cv = sd(true) / mean(true)) %>%
  summarize(mean(bayes_cv / direct_cv)) %>%
  pull()
```

```
## 'summarise()' ungrouping output (override with '.groups' argument)
```

```
## [1] 0.6661926
```

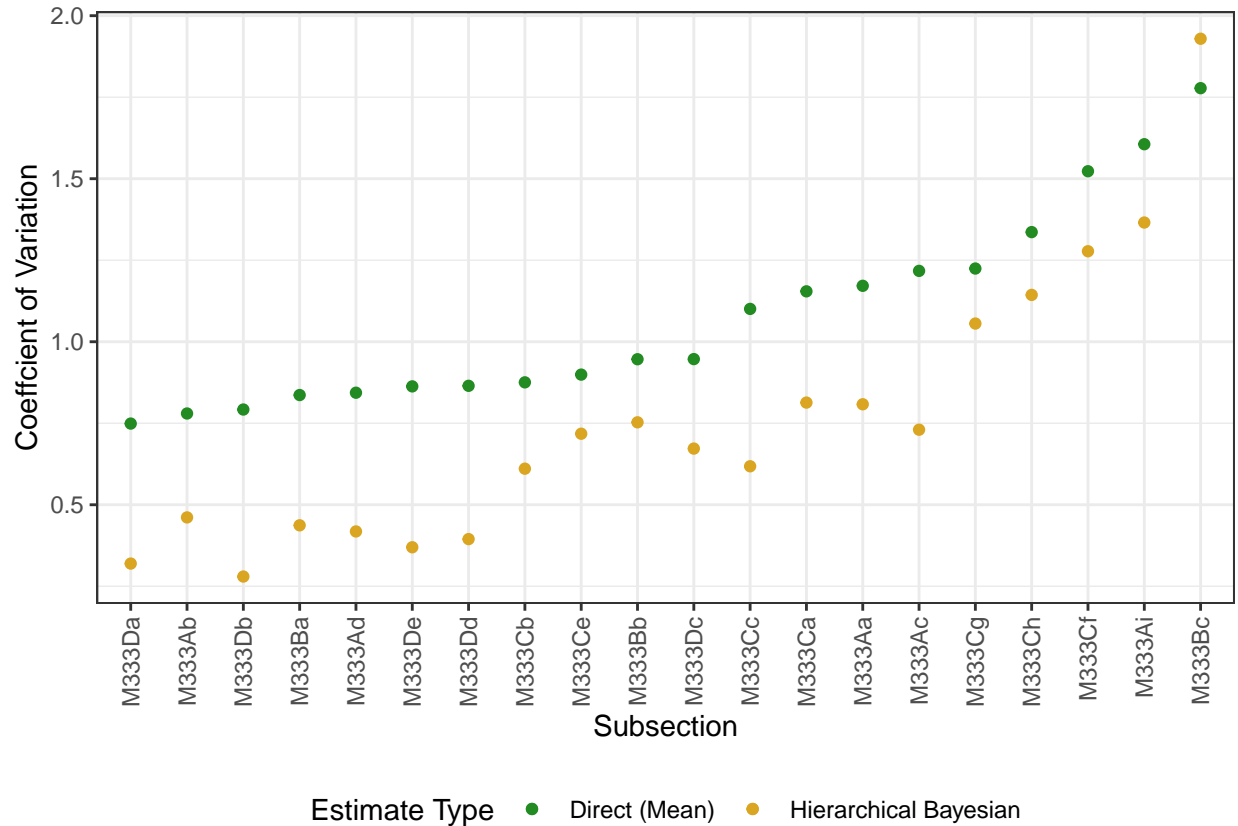
```
mod_df %>%
  group_by(subsection) %>%
  summarize(bayes_cv = sd(fitted) / mean(fitted),
            direct_cv = sd(true) / mean(true)) %>%
  mutate(subsection = fct_reorder(subsection, direct_cv)) %>%
  ggplot(aes(x = subsection)) +
  geom_point(aes(y = bayes_cv, color = "goldenrod"), ) +
  geom_point(aes(y = direct_cv, color = "forestgreen")) +
  theme_bw() +
  scale_color_manual(
    name = 'Estimate Type',
    values = c('goldenrod' = 'goldenrod',
               'forestgreen' = 'forestgreen'),
    labels = c('Direct (Mean)', 'Hierarchical Bayesian'),
    guide = "legend"
  ) +
  theme(axis.text.x = element_text(
    angle = 90,
```

```

  vjust = 0.5,
  hjust = 1
),
legend.position = "bottom") +
labs(x = "Subsection",
     y = "Coefficient of Variation")

```

```
## 'summarise()' ungrouping output (override with '.groups' argument)
```



```

mod_df %>%
  group_by(subsection) %>%
  summarize(mean_direct = mean(true),
            mean_bayes = mean(fitted)) %>%
  mutate(subsection = fct_reorder(subsection, mean_direct)) %>%
  ggplot(aes(x = subsection)) +
  geom_point(aes(y = mean_bayes,
                 color = "goldenrod"),
            position = position_nudge(x = -0.1)) +
  geom_point(aes(y = mean_direct,
                 color = "forestgreen"),
            position = position_nudge(x = 0.1)) +
  theme_bw() +
  scale_color_manual(
    name = 'Estimate Type',

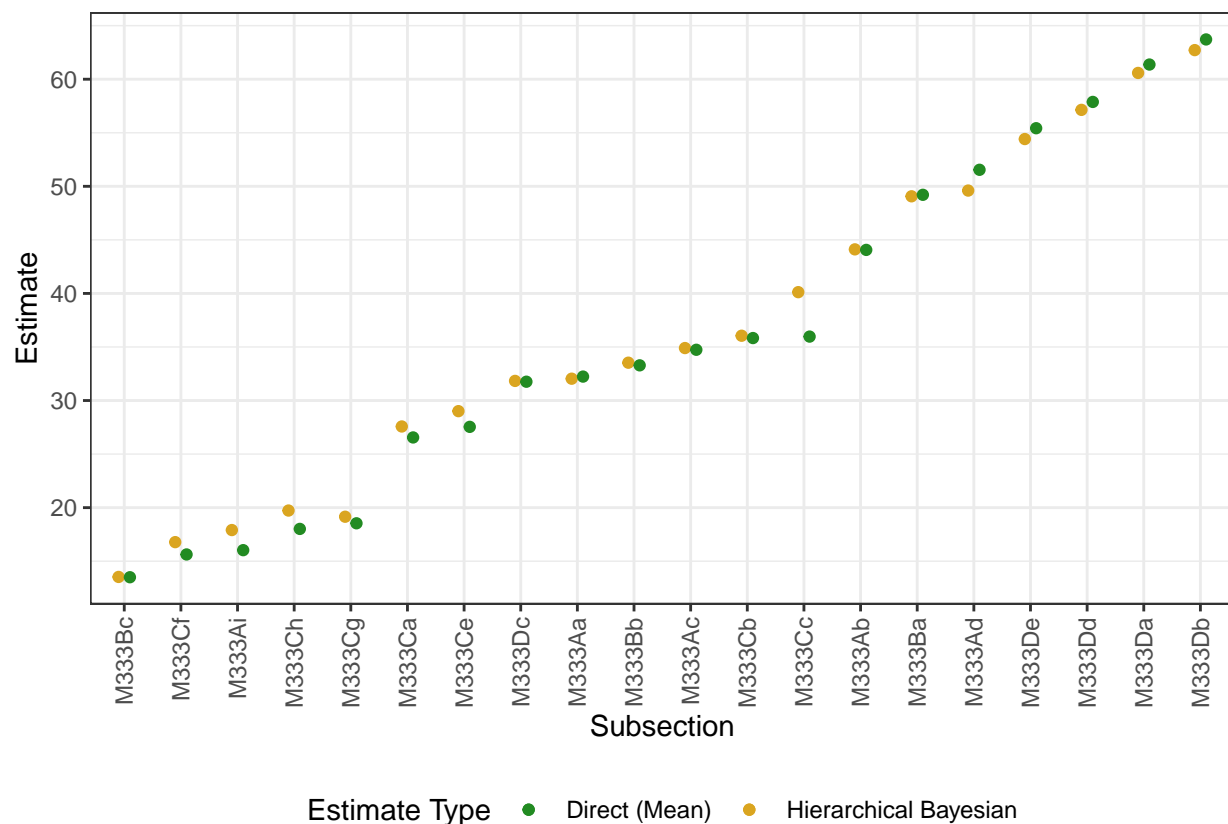
```

```

values = c('goldenrod' = 'goldenrod',
           'forestgreen' = 'forestgreen'),
labels = c('Direct (Mean)', 'Hierarchical Bayesian'),
guide = "legend"
) +
theme(axis.text.x = element_text(
  angle = 90,
  vjust = 0.5,
  hjust = 1
),
legend.position = "bottom") +
labs(x = "Subsection",
     y = "Estimate")

```

```
## 'summarise()' ungrouping output (override with '.groups' argument)
```



All the provinces

```
df <- read_csv("../data/subsets/dat_small.csv")
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   subsection = col_character(),
##   section = col_character(),
##   province = col_character()
## )

## See spec(...) for full column specifications.
```

```
df <- df %>%
  mutate(id = province) %>%
  group_by(id) %>%
  nest()

models_df <- list()
cv <- c()
models <- readRDS("models.rds")
for(i in 1:14) {
  # models[[i]] <- stan_lmer(BIOLIVE_TPA ~ 1 + nlcd11 + (1 | subsection),
  #                           data = df[[i,2]][[1]],
  #                           verbose = FALSE)

  models_df[[i]] <- data.frame(
    fitted = models[[i]]$fitted.values,
    subsection = df[[i,2]][[1]] %>% dplyr::select(subsection),
    true = models[[i]]$y
  ) }

cv <- list()
for(i in 1:14){

cv[i] <- models_df[[i]] %>%
  group_by(subsection) %>%
  summarize(bayes_cv = sd(fitted, na.rm = TRUE) / mean(fitted, na.rm = TRUE),
            direct_cv = sd(true, na.rm = TRUE) / mean(true, na.rm = TRUE)) %>%
  summarize(median(bayes_cv / direct_cv, na.rm = TRUE))
}
```

```
## 'summarise()' ungrouping output (override with '.groups' argument)

## 'summarise()' ungrouping output (override with '.groups' argument)
## 'summarise()' ungrouping output (override with '.groups' argument)
## 'summarise()' ungrouping output (override with '.groups' argument)
## 'summarise()' ungrouping output (override with '.groups' argument)
## 'summarise()' ungrouping output (override with '.groups' argument)
## 'summarise()' ungrouping output (override with '.groups' argument)
## 'summarise()' ungrouping output (override with '.groups' argument)
## 'summarise()' ungrouping output (override with '.groups' argument)
## 'summarise()' ungrouping output (override with '.groups' argument)
## 'summarise()' ungrouping output (override with '.groups' argument)
```

```
## 'summarise()' ungrouping output (override with '.groups' argument)
## 'summarise()' ungrouping output (override with '.groups' argument)
```

```
unlist(cv)
```

```
## [1] 0.7874218 0.6088884 0.7469403 0.2930950 0.7840197 0.7830745 0.6622711
## [8] 0.6516997 0.7034978 0.6956291 0.6941903 0.6490448 0.9724442 0.7304559
```

```
mean(unlist(cv))
```

```
## [1] 0.6973338
```