

```
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 0.000362 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 3.62 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: 7.98935 seconds (Warm-up)
## Chain 1:                3.752 seconds (Sampling)
## Chain 1:                11.7414 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0.000157 seconds
```

```

## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 1.57 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: 9.02879 seconds (Warm-up)
## Chain 2:                3.696 seconds (Sampling)
## Chain 2:                12.7248 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0.000141 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 1.41 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: 8.11648 seconds (Warm-up)
## Chain 3:                3.09289 seconds (Sampling)
## Chain 3:                11.2094 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0.000154 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 1.54 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: 8.52788 seconds (Warm-up)
## Chain 4:                3.42436 seconds (Sampling)
## Chain 4:                11.9522 seconds (Total)
## Chain 4:
```

```
# ggplot(m333, aes(x = nlcd11,
#                  y = BIOLIVE_TPA)) +
#   geom_point() +
#   facet_wrap(~subsection)
```

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

```
mod_df %>%
  yardstick::metrics(truth = true,
                     estimate = fitted)
```

```
## # A tibble: 3 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>      <dbl>
## 1 rmse    standard      29.4
## 2 rsq     standard       0.461
## 3 mae     standard       20.7
```

```
sub_mean <- mod_df %>%
  group_by(subsection) %>%
  summarize(mean_y = mean(true))
```

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

```
mod_df %>%
  group_by(subsection) %>%
  yardstick::rmse(truth = true,
                  estimate = fitted) %>%
  left_join(sub_mean) %>%
  summarize(cv = .estimate / mean_y)
```

```
## Joining, by = "subsection"
```

```
## # A tibble: 20 x 1
```

```
##       cv
```

```
##   <dbl>
```

```
## 1 0.812
```

```
## 2 0.677
```

```
## 3 0.975
```

```
## 4 0.607
```

```
## 5 1.08
```

```
## 6 0.652
```

```
## 7 0.708
```

```
## 8 1.19
```

```
## 9 0.810
```

```
## 10 0.670
```

```
## 11 0.888
```

```
## 12 0.710
```

```
## 13 1.05
```

```
## 14 0.742
```

```
## 15 0.957
```

```
## 16 0.609
```

```
## 17 0.694
```

```
## 18 0.743
```

```
## 19 0.711
```

```
## 20 0.728
```

```
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()
```

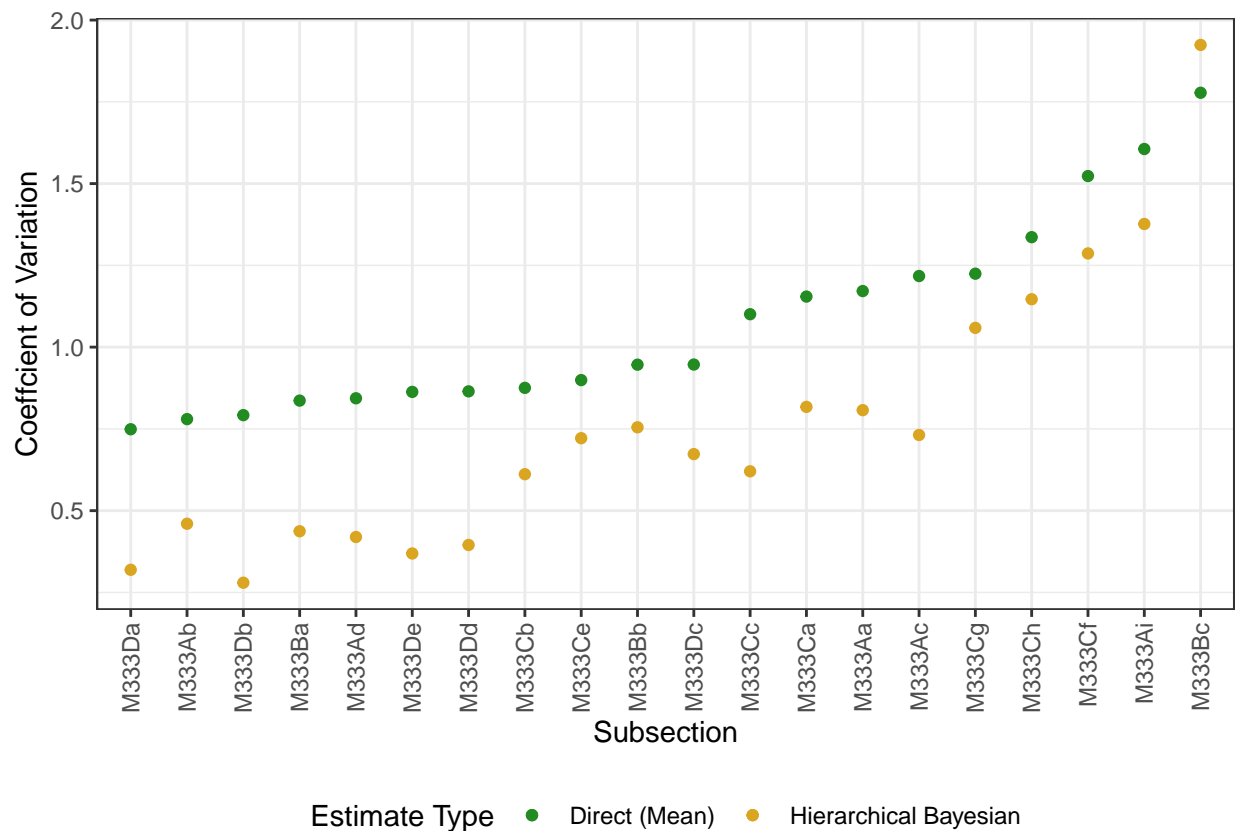
```
## 'summarise()' ungrouping output (override with '.groups' argument)
```

```
## [1] 0.6675408
```

```
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)
```



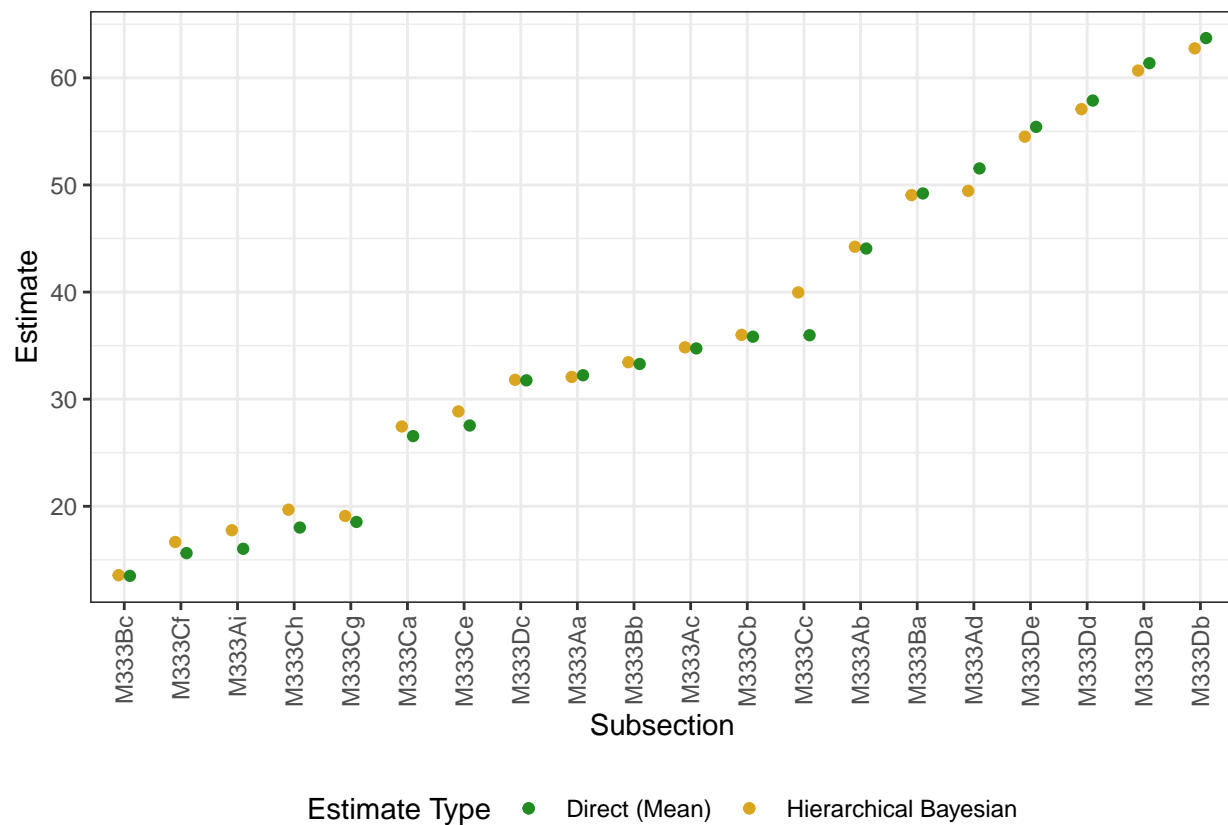
```
mod_df %>%
  group_by(subsection) %>%
  summarize(mean_direct = mean(true),
            mean_bayes = mean(fitted)) %>%
  mutate(subsection = fct_reorder(subsection, mean_direct)) %>%
  ggplot(aes(x = subsection)) +
  geom_point(aes(y = mean_bayes,
                 color = "goldenrod"),
            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)
## '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
```