FixedReduced2_SimulationStudy

Syed Rafey Abbas

2024-07-17

```
# Defining the BIC functions, looking at three definitions
bic_fitzmaurice <- function(model) {
  logLik_val <- as.numeric(logLik(model))
  n_params <- length(fixef(model)) + sum(sapply(VarCorr(model), function(x) prod(dim(x))))
  N <- length(unique(model@frame$group_id)) # Number of subjects
  BIC_value <- -2 * logLik_val + log(N) * n_params
  return(BIC_value)
}

bic_normal <- function(model) {
  logLik_val <- as.numeric(logLik(model))
  n_params <- length(fixef(model)) + sum(sapply(VarCorr(model), function(x) prod(dim(x))))
  N <- nrow(model@frame) # Total number of observations
  BIC_value <- -2 * logLik_val + log(N) * n_params
  return(BIC_value)
}

bic_hybrid <- function(model) {
  logLik_val <- as.numeric(logLik(model))
  n_fixed <- length(fixef(model))
  n_random <- sum(sapply(VarCorr(model), function(x) prod(dim(x))))
  N <- nrow(model@frame) # Total number of observations
  m <- length(unique(model@frame$group_id)) # Number of groups
  BIC_value <- -2 * logLik_val + n_fixed * log(N) + n_random * log(m)
  return(BIC_value)
}
```

```
# Data generation function
generate_data <- function(ni, m, beta, random_effects_var) {
  group_id <- rep(1:m, each = ni)
  x1 <- rnorm(ni * m)
  x2 <- rnorm(ni * m)
  x3 <- rnorm(ni * m)
  epsilon <- rnorm(ni * m)
  random_effect <- rep(rnorm(m, mean = 0, sd = sqrt(random_effects_var)), each = ni)
  y <- beta[1] * x1 + beta[3] * x3 + rep(random_effect, each = ni) + epsilon

  data <- data.frame(y = y, x1 = x1, x2 = x2, x3 = x3, group_id = factor(group_id))
  return(data)
}

# Model fitting function
```

```r
fit_models <- function(data) {
  models <- list(
    full = lmer(y ~ x1 + x2 + x3 + (1|group_id), data = data),
    reduced1 = lmer(y ~ x1 + x2 + (1|group_id), data = data),
    reduced2 = lmer(y ~ x1 + x3 + (1|group_id), data = data),
    reduced3 = lmer(y ~ x2 + x3 + (1|group_id), data = data)
  )
  return(models)
}

# BIC calculation function
calculate_bic <- function(models, bic_func) {
  bic_values <- sapply(models, bic_func)
  return(bic_values)
}

# Simulate and fit models
simulate_and_fit_models <- function(ni, m, beta, random_effects_var, i) {
  message("Starting simulation ", i)
  data <- generate_data(ni, m, beta, random_effects_var)
  models <- fit_models(data)

  bic_fitzmaurice_values <- calculate_bic(models, bic_fitzmaurice)
  bic_normal_values <- calculate_bic(models, bic_normal)
  bic_hybrid_values <- calculate_bic(models, bic_hybrid)

  selected_model_fitzmaurice <- names(which.min(bic_fitzmaurice_values))
  selected_model_normal <- names(which.min(bic_normal_values))
  selected_model_hybrid <- names(which.min(bic_hybrid_values))

  results_df <- data.frame(
    true_model = "reduced2",
    model_name = names(models),
    bic_fitzmaurice = bic_fitzmaurice_values,
    bic_normal = bic_normal_values,
    bic_hybrid = bic_hybrid_values,
    selected_model_fitzmaurice = selected_model_fitzmaurice,
    selected_model_normal = selected_model_normal,
    selected_model_hybrid = selected_model_hybrid,
    correct_model_fitzmaurice = selected_model_fitzmaurice == "reduced2",
    correct_model_normal = selected_model_normal == "reduced2",
    correct_model_hybrid = selected_model_hybrid == "reduced2"
  )

  message("Ending simulation ", i)
  return(results_df)
}
```

```r
# Function to run multiple simulations
run_simulation <- function(num_simulations, ni, m, beta, random_effects_var, ncores = detectCores()) {
  cat("Running simulations...\n")
  results <- mclapply(1:num_simulations, function(i) {
    simulate_and_fit_models(ni, m, beta, random_effects_var, i)
```

```
  }, mc.cores = ncores)

  return(do.call(rbind, results))
}

# Modified function to run simulations for different subject numbers
run_simulations_for_subject_numbers <- function(subject_numbers, num_simulations, m, beta, random_effec
  results_list <- list()

  for (ni in subject_numbers) {
    cat(sprintf("\nRunning simulations for subject number ni = %d...\n", ni))
    results <- run_simulation(num_simulations, ni, m, beta, random_effects_var, ncores = ncores)
    results <- na.omit(results)
    results_list[[paste0("ni_", ni)]] <- results
  }

  return(results_list)
}
```

```
# Parameters for the simulation
set.seed(123)
num_simulations <- 100
m <- 30
beta <- c(1, 0, 1)
random_effects_var <- 1
subject_numbers <- c(10, 30, 50)

# Run simulations and save results
results <- run_simulations_for_subject_numbers(subject_numbers, num_simulations, m, beta, random_effects
saveRDS(results, file = "simulation_results_list.rds")
```

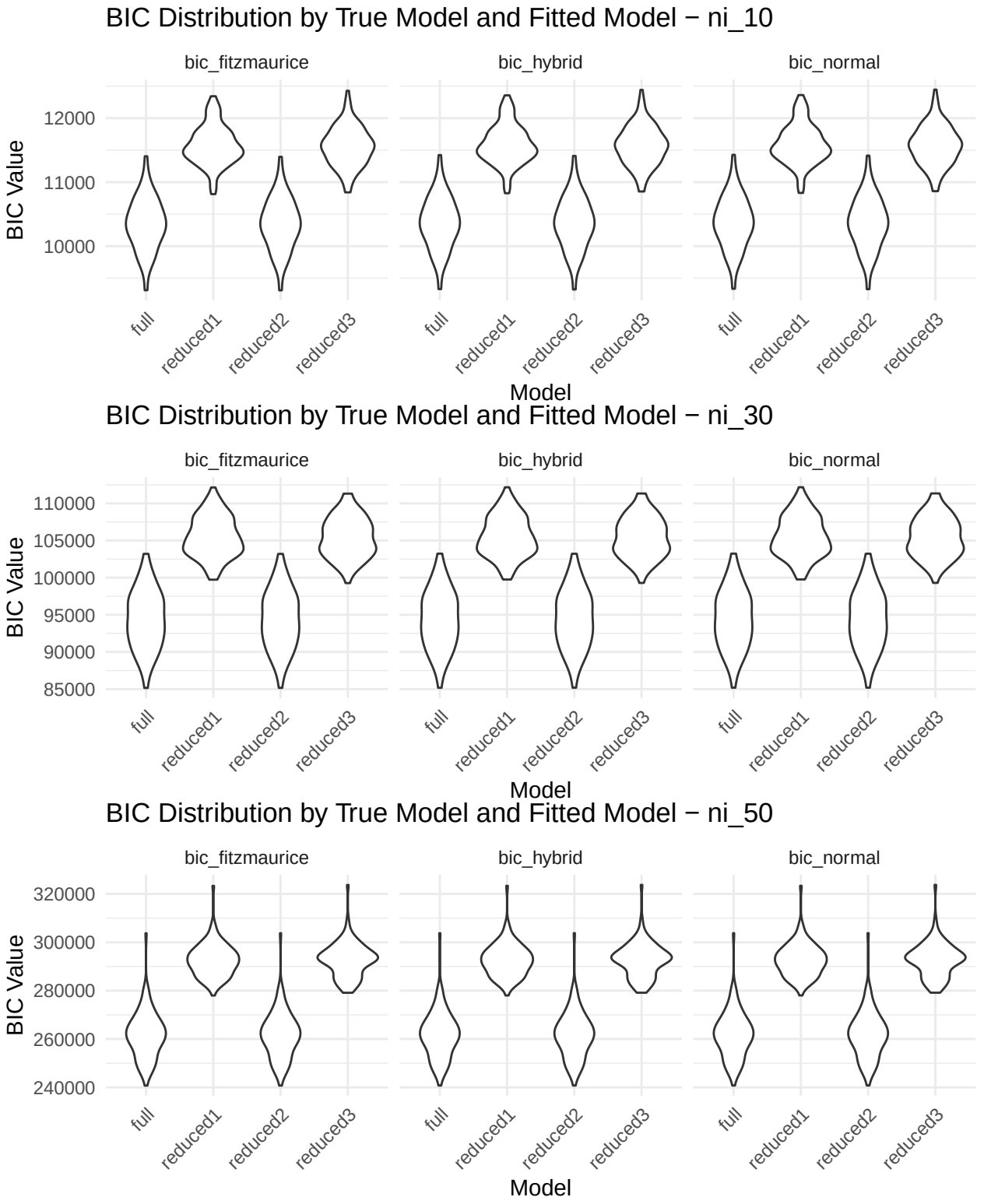
```
results_list <- readRDS("simulation_results_list.rds")

# Initialize lists to store plots
violin_plots <- list()
bar_plots <- list()

for (name in names(results_list)) {
  all_results <- results_list[[name]]

  # Convert BIC columns to long format for plotting
  bics_long <- pivot_longer(all_results, cols = starts_with("bic_"), names_to = "BIC_definition", value

  # Convert selected model columns to long format for plotting
  selected_models_long <- pivot_longer(all_results, cols = starts_with("selected_model_"),
                                       names_to = "BIC_definition", values_to = "selected_model")

  # Violin plot for BIC values
  violin_plot <- ggplot(bics_long, aes(x = model_name, y = BIC)) +
    geom_violin() +
    facet_wrap(~ BIC_definition) +
    theme_minimal() +
    labs(title = paste("BIC Distribution by True Model and Fitted Model -", name),
```

```
        x = "Model", y = "BIC Value") +
    theme(axis.text.x = element_text(angle = 45, hjust = 1),
          plot.margin = margin(0, 0, 0, 0))

  violin_plots[[name]] <- violin_plot

  # Bar plot for selected models
  bar_plot <- ggplot(selected_models_long, aes(x = BIC_definition, fill = selected_model)) +
    geom_bar(position = "fill", width = 1) +  # Make sure bars are adjacent
    facet_wrap(~ BIC_definition) +
    theme_minimal() +
    labs(title = paste("Model Selection Proportions by True Model -", name),
         x = "True Model", y = "Proportion",
         fill = "Selected Model") +
    theme(axis.text.x = element_text(angle = 45, hjust = 1),
          plot.margin = margin(0, 0, 0, 0))

  bar_plots[[name]] <- bar_plot

  # Print summary
  summary_table <- selected_models_long %>%
    group_by(BIC_definition, true_model, selected_model) %>%
    summarise(count = n(), .groups = 'drop') %>%
    pivot_wider(names_from = selected_model, values_from = count, values_fill = list(count = 0))

  print(paste("Summary for", name))
  print(summary_table)
}
```

```
## [1] "Summary for ni_10"
## # A tibble: 3 x 4
##   BIC_definition             true_model  full reduced2
##   <chr>                      <chr>      <int>    <int>
## 1 selected_model_fitzmaurice reduced2      92      308
## 2 selected_model_hybrid      reduced2      72      328
## 3 selected_model_normal      reduced2      72      328
## [1] "Summary for ni_30"
## # A tibble: 3 x 4
##   BIC_definition             true_model  full reduced2
##   <chr>                      <chr>      <int>    <int>
## 1 selected_model_fitzmaurice reduced2     164      236
## 2 selected_model_hybrid      reduced2     104      296
## 3 selected_model_normal      reduced2     104      296
## [1] "Summary for ni_50"
## # A tibble: 3 x 4
##   BIC_definition             true_model  full reduced2
##   <chr>                      <chr>      <int>    <int>
## 1 selected_model_fitzmaurice reduced2     196      204
## 2 selected_model_hybrid      reduced2     156      244
## 3 selected_model_normal      reduced2     156      244
```

```
# Combine all violin plots using patchwork
combined_violin_plot <- wrap_plots(violin_plots, ncol = 1) & theme(plot.margin = margin(0, 0, 0, 0))
```

```r
combined_bar_plot <- wrap_plots(bar_plots, ncol = 1) & theme(plot.margin = margin(0, 0, 0, 0))

# Display combined plots
print(combined_violin_plot)
```



```
#print(combined_bar_plot)
```

BIC proportion plot

```
all_results <- results_list %>% bind_rows(.id = "ni")
row.names(all_results) <- NULL

selected_models_long <- pivot_longer(all_results,
                                     cols =starts_with("selected_model_"), names_to = "BIC_definition",
  mutate(BIC_definition = factor(BIC_definition,
                                 levels = c("selected_model_fitzmaurice", "selected_model_normal", "sele
                                 labels = c("BIC_f", "BIC_s","BIC_h")))

ggplot(selected_models_long, aes(x = BIC_definition, fill = selected_model)) +
    geom_bar() +
    facet_wrap(~ ni) +
    theme_minimal() +
    labs(title = "Model Selection Proportions",
         x = "", y = "Proportion",
         fill = "Selected Model") #+
```


```
    #theme(axis.text.x = element_text(angle = 45, hjust = 1),
    #      plot.margin = margin(0, 0, 0, 0))
```