Case Study for "Potential bias models with Bayesian shrinkage priors for dynamic borrowing of multiple historical control data''

Tomohiro Ohigashi, Kazushi Maruo, Takashi Sozu, Ryo Sawamoto, Masahiko Gosho

Contents

1 Required R packages

```
library(tidyverse)
library(ggpubr)
library(rstan)
library(cmdstanr)
library(bayesplot)
library(RBesT)
options(kableExtra.latex.load_packages = FALSE)
library(kableExtra)
library(knitr)
```

2 Preparation

2.1 Reading Stan Code

All Stan codes used in the case study (and simulation studies) are stored in the "Stan'' directory. In the following, these codes are read.

```
Current_stan <- cmdstan_model(paste0(getwd(), '/Stan/Current.stan'))
Pooled_stan  <- cmdstan_model(paste0(getwd(), '/Stan/Pooled.stan'))
EX_stan      <- cmdstan_model(paste0(getwd(), '/Stan/EX.stan'))
EXNEX_stan   <- cmdstan_model(paste0(getwd(), '/Stan/EXNEX.stan'))
DPMEX_stan   <- cmdstan_model(paste0(getwd(), '/Stan/DPMEX.stan'))
DMPP_stan    <- cmdstan_model(paste0(getwd(), '/Stan/DMPP.stan'))
UIPDir_stan  <- cmdstan_model(paste0(getwd(), '/Stan/UIPDir.stan'))
HS_stan      <- cmdstan_model(paste0(getwd(), '/Stan/HS.stan'))
DL_stan      <- cmdstan_model(paste0(getwd(), '/Stan/DL.stan'))
SS_stan      <- cmdstan_model(paste0(getwd(), '/Stan/SS.stan'))
SSL_stan     <- cmdstan_model(paste0(getwd(), '/Stan/SSL.stan'))
```

2.2 Create R function for calculating effective sample size and drawing posterior distributions for β_{H_k}

```
ess_calc3 <- function(fit){
  mix <- mixfit(fit$draws("pi_C"), type = "beta",
                Nc = 3, constrain_gt1 =TRUE, eps = 0.01)
  ESS <- ess(mix)
  return(ESS)
}


beta_fig <- function(fit, titlevar){
  result_fit <- fit$output_files() %>% rstan::read_stan_csv()
  plot100 <- stan_dens(result_fit, pars = c("beta"))$data
  plot100_ <- plot100[order(plot100$parameter, plot100$chain),]
  plot100_1 <- plot100_ %>% filter(parameter %in%
      c("beta[1]", "beta[2]", "beta[3]", "beta[4]", "beta[5]",
        "beta[6]", "beta[7]", "beta[8]" )) %>%
    mutate(color = case_when(
      parameter == "beta[1]" ~ "beta1",
      parameter == "beta[2]" ~ "beta2",
      parameter == "beta[3]" ~ "beta3",
      parameter == "beta[4]" ~ "beta4",
      parameter == "beta[5]" ~ "beta5",
      parameter == "beta[6]" ~ "beta6",
      parameter == "beta[7]" ~ "beta7",
      parameter == "beta[8]" ~ "beta8",
      TRUE ~ " "
    )
    )

  out <- ggplot(data = plot100_1, aes(x = value, color = color)) +
```

```
    labs(title = titlevar) +
    stat_density(geom = "line", position = "identity", size = 12) +
    scale_colour_manual(
      "beta",
      values = c("#03A9F4", "#1976D2", "#4E944F", "#125B50",
                 "#D4D925", "#E8630A", "#FF0099", "#9900F0"),
      labels = c(expression(beta[1]), expression(beta[2]),
                 expression(beta[3]), expression(beta[4]),
                 expression(beta[5]), expression(beta[6]),
                 expression(beta[7]), expression(beta[8]))
    ) +
    xlab(expression(beta)) +
    ylab("Density") +
    scale_x_continuous(breaks = seq(-2, 4, by = 2), limits = c(-2, 4)) +
    scale_y_continuous(breaks = seq(0, 2, by = 1), limits = c(0, 2.5)) +
    theme(
      plot.title = element_text(size = 240),
      axis.text = element_text(colour = "black", size = 240),
      axis.ticks=element_line(colour = "black", size = 1),
      axis.title.x = element_text(size = 240),
      axis.title.y = element_text(size = 240),
      axis.line = element_line(colour = "#000000",
                    size = 10, linetype = "solid", lineend = "round"),
      panel.background = element_rect(fill = "transparent", colour = NA),
      panel.border = element_blank(),
      panel.grid = element_blank(),
      panel.grid.major = element_blank(),
      panel.grid.minor = element_blank(),
      panel.grid.major.x = element_blank(),
      panel.grid.major.y = element_blank(),
      panel.grid.minor.x = element_blank(),
      panel.grid.minor.y = element_blank(),
      legend.title = element_blank(),
      legend.text = element_text(size = 240),
      legend.key = element_rect(fill = "transparent", colour = NA),
      legend.key.width = unit(20, "cm"),
      legend.position = c(0.8, 0.5),
      legend.key.size = unit(2, 'lines'),
      legend.spacing.x = unit(10, 'cm')
    )
  return(out)
}
```

3 Case study 1 (original data)

3.1 Datasets

```
n_h <- c(107, 44, 51, 39, 139, 20, 78, 35)
x_h <- c( 23, 12, 19,  9,  39,  6,  9, 10)
```

```
n_CC <- 6
x_CC <- 1
n_CT <- 23
x_CT <- 14

Historical <- data.frame(
  N = n_h,
  X = x_h
)

Current <- data.frame(
  TRT = c(0, 1),
  N = c(n_CC, n_CT),
  X = c(x_CC, x_CT)
)

Data_all <- list(Historical = Historical, Current = Current)
ANA <- list(H = length(Data_all$Historical[,"N"]),
            x_h = Data_all$Historical[,"X"], n_h = Data_all$Historical[,"N"],
            x_CC = Data_all$Current$X[1], n_CC = Data_all$Current$N[1],
            x_CT = Data_all$Current$X[2], n_CT = Data_all$Current$N[2],
            betascale = 1, nu = 1, C = 10)
```

3.2 MCMC fitting

```
fit_Current  <- Current_stan$sample(data = ANA, seed = 999, chains = 4,
        parallel_chains = 4, iter_warmup = 4000, iter_sampling = 4000, refresh = 0)

fit_Pooled <- Pooled_stan$sample(data = ANA, seed = 999, chains = 4,
        parallel_chains = 4, iter_warmup = 4000, iter_sampling = 4000, refresh = 0)

fit_EX     <- EX_stan$sample(data = ANA, seed = 999, chains = 4,
        parallel_chains = 4, iter_warmup = 4000, iter_sampling = 4000,
        adapt_delta = 0.9, refresh = 0)

fit_EXNEX  <- EXNEX_stan$sample(data = ANA, seed = 999, chains = 4,
        parallel_chains = 4, iter_warmup = 4000, iter_sampling = 4000,
        adapt_delta = 0.9, refresh = 0)

fit_DPMEX  <- DPMEX_stan$sample(data = ANA, seed = 999, chains = 4,
        parallel_chains = 4, iter_warmup = 4000, iter_sampling = 6000,
        adapt_delta = 0.95, refresh = 0)

fit_DMPP   <- DMPP_stan$sample(data = ANA, seed = 999, chains = 4,
        parallel_chains = 4, iter_warmup = 4000, iter_sampling = 6000,
        adapt_delta = 0.95, refresh = 0)

gamma_h = rep(NA, length(Data_all$Historical[,"N"]))
for(h in 1:length(Data_all$Historical[,"N"])){
```

```r
  gamma_h[h] = min(1, Data_all$Historical[,"N"][h] / Data_all$Current$N[1])
}

theta_h <- rep(NA, length(n_h))
I_U <- rep(NA, length(n_h))
for(h in 1:length(n_h)){
  theta_h[h] = x_h[h] * 1.0 / n_h[h]
  I_U[h] = 1 / (theta_h[h] * (1 - theta_h[h]))
}

ANA_UIP <- list(H = length(Data_all$Historical[,"N"]),
                x_h = Data_all$Historical$X, n_h = Data_all$Historical$N,
                x_CC = Data_all$Current$X[1], n_CC = Data_all$Current$N[1],
                x_CT = Data_all$Current$X[2], n_CT = Data_all$Current$N[2],
                gamma_h = gamma_h, theta_h = theta_h, I_U = I_U)

fit_UIPDir  <- UIPDir_stan$sample(data = ANA_UIP, seed = 999, chains = 4,
        parallel_chains = 4, iter_warmup = 4000, iter_sampling = 8000,
        adapt_delta = 0.999, max_treedepth = 12, refresh = 0)


fit_HS     <- HS_stan$sample(data = ANA, seed = 999, chains = 4,
        parallel_chains = 4, iter_warmup = 4000, iter_sampling = 12000,
        adapt_delta = 0.999, refresh = 0)

ANA_DL <- list(H = length(Data_all$Historical[,"N"]),
               x_h = Data_all$Historical[,"X"], n_h = Data_all$Historical[,"N"],
               x_CC = Data_all$Current$X[1], n_CC = Data_all$Current$N[1],
               x_CT = Data_all$Current$X[2], n_CT = Data_all$Current$N[2], a = 0.5)

fit_DL <- DL_stan$sample(data = ANA_DL, seed = 999, chains = 4,
        parallel_chains = 4, iter_warmup = 4000, iter_sampling = 4000,
        adapt_delta = 0.9, refresh = 0)

fit_SS     <- SS_stan$sample(data = ANA, seed = 999, chains = 4,
        parallel_chains = 4, iter_warmup = 4000, iter_sampling = 6000,
        adapt_delta = 0.95, max_treedepth = 12, refresh = 0)

ANA_SSL <- list(H = length(Data_all$Historical[,"N"]),
                x_h = Data_all$Historical[,"X"], n_h = Data_all$Historical[,"N"],
                x_CC = Data_all$Current$X[1], n_CC = Data_all$Current$N[1],
                x_CT  =Data_all$Current$X[2], n_CT = Data_all$Current$N[2],
                lambda_spike = 0.1, lambda_slab = 2.0)

fit_SSL <- SSL_stan$sample(data = ANA_SSL, seed = 999, chains = 4,
        parallel_chains = 4, iter_warmup = 4000, iter_sampling = 4000,
        adapt_delta = 0.9, refresh = 0)
```

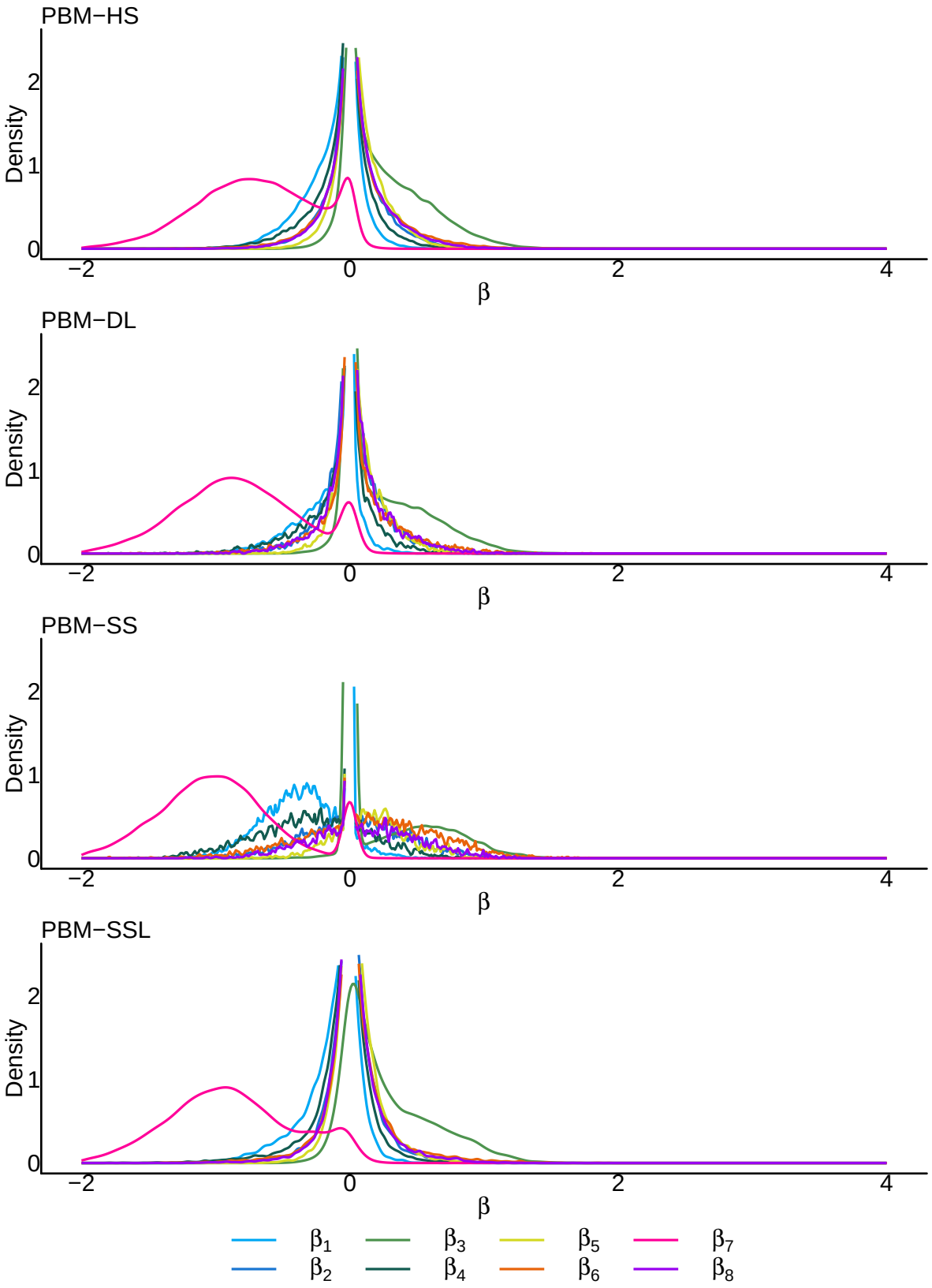
3.3 Aggregate posterior distribution

```
Result_ALL <- matrix(0, nrow = 11, ncol = 7)
Result_ALL[1,]  <- c(1, mean(fit_Current$draws("g_pi")), sd(fit_Current$draws("g_pi")),
                     quantile(fit_Current$draws("g_pi"), 0.025, names = F),
                     quantile(fit_Current$draws("g_pi"), 0.975, names = F),
                     max(fit_Current$summary()$rhat), NA)
Result_ALL[2,]  <- c(2, mean(fit_Pooled$draws("g_pi")), sd(fit_Pooled$draws("g_pi")),
                     quantile(fit_Pooled$draws("g_pi"), 0.025, names = F),
                     quantile(fit_Pooled$draws("g_pi"), 0.975, names = F),
                     max(fit_Pooled$summary()$rhat), NA)
Result_ALL[3,]  <- c(3, mean(fit_EX$draws("g_pi")), sd(fit_EX$draws("g_pi")),
                     quantile(fit_EX$draws("g_pi"), 0.025, names = F),
                     quantile(fit_EX$draws("g_pi"), 0.975, names = F),
                     max(fit_EX$summary()$rhat), ess_calc3(fit_EX))
Result_ALL[4,]  <- c(4, mean(fit_EXNEX$draws("g_pi")), sd(fit_EXNEX$draws("g_pi")),
                     quantile(fit_EXNEX$draws("g_pi"), 0.025, names = F),
                     quantile(fit_EXNEX$draws("g_pi"), 0.975, names = F),
                     max(fit_EXNEX$summary()$rhat), ess_calc3(fit_EXNEX))
Result_ALL[5,]  <- c(5, mean(fit_DPMEX$draws("g_pi")), sd(fit_DPMEX$draws("g_pi")),
                     quantile(fit_DPMEX$draws("g_pi"), 0.025, names = F),
                     quantile(fit_DPMEX$draws("g_pi"), 0.975, names = F),
                     max(fit_DPMEX$summary()$rhat), ess_calc3(fit_DPMEX))
Result_ALL[6,]  <- c(6, mean(fit_DMPP$draws("g_pi")), sd(fit_DMPP$draws("g_pi")),
                     quantile(fit_DMPP$draws("g_pi"), 0.025, names = F),
                     quantile(fit_DMPP$draws("g_pi"), 0.975, names = F),
                     max(fit_DMPP$summary()$rhat), ess_calc3(fit_DMPP))
Result_ALL[7,]  <- c(7, mean(fit_UIPDir$draws("g_pi")), sd(fit_UIPDir$draws("g_pi")),
                     quantile(fit_UIPDir$draws("g_pi"), 0.025, names = F),
                     quantile(fit_UIPDir$draws("g_pi"), 0.975, names = F),
                     max(fit_UIPDir$summary()$rhat), ess_calc3(fit_UIPDir))
Result_ALL[8,]  <- c(8, mean(fit_HS$draws("g_pi")), sd(fit_HS$draws("g_pi")),
                     quantile(fit_HS$draws("g_pi"), 0.025, names = F),
                     quantile(fit_HS$draws("g_pi"), 0.975, names = F),
                     max(fit_HS$summary()$rhat) , ess_calc3(fit_HS))
Result_ALL[9,]  <- c(9, mean(fit_DL$draws("g_pi")), sd(fit_DL$draws("g_pi")),
                     quantile(fit_DL$draws("g_pi"), 0.025, names = F),
                     quantile(fit_DL$draws("g_pi"), 0.975, names = F),
                     max(fit_DL$summary()$rhat), ess_calc3(fit_DL))
Result_ALL[10,]  <- c(10, mean(fit_SS$draws("g_pi")), sd(fit_SS$draws("g_pi")),
                      quantile(fit_SS$draws("g_pi"), 0.025, names = F),
                      quantile(fit_SS$draws("g_pi"), 0.975, names = F),
                      max(fit_SS$summary()$rhat), ess_calc3(fit_SS))
Result_ALL[11,]  <- c(11, mean(fit_SSL$draws("g_pi")), sd(fit_SSL$draws("g_pi")),
                      quantile(fit_SSL$draws("g_pi"), 0.025, names = F),
                      quantile(fit_SSL$draws("g_pi"), 0.975, names = F),
                      max(fit_SSL$summary()$rhat), ess_calc3(fit_SSL))

Result_ALL1 <- as.data.frame(Result_ALL, stringsAsFactors = F)
colnames(Result_ALL1) <- c("MethodN", "Mean", "SD", "LCI", "UCI", "diag", "ESS")
```

```r
Result_ALL1 <- Result_ALL1 %>% mutate(Range = (UCI - LCI)) %>%
  mutate(Method = case_when(
    MethodN == 1 ~ "CD",
    MethodN == 2 ~ "PD",
    MethodN == 3 ~ "EX",
    MethodN == 4 ~ "EXNEX",
    MethodN == 5 ~ "DPMEX",
    MethodN == 6 ~ "DMPP",
    MethodN == 7 ~ "UIP",
    MethodN == 8 ~ "PBM-HS",
    MethodN == 9 ~ "PBM-DL",
    MethodN == 10 ~ "PBM-SS",
    MethodN == 11 ~ "PBM-SSL",
    TRUE ~ ""
  )) %>%
  mutate(Mean1 = Mean * 100, SD1 = SD * 100,
         LCI1 = LCI * 100, UCI1 = UCI * 100,
         CI1 = paste0("(", format(round(LCI * 100, digits = 1), nsmall = 1), ", ",
                      format(round(UCI * 100, digits = 1), nsmall = 1), ")")) %>%
  mutate(EHSS1 = if_else(MethodN %in% c(1, 2), "",
                         format(round(ESS - n_CC, digits = 1), nsmall = 1))) %>%
  select(c(Method, Mean1, SD1, LCI1, UCI1, CI1, EHSS1)) %>%
  mutate(case = "1")

Result_ALL1$Method <- factor(Result_ALL1$Method,
        levels = rev(c("CD", "PD", "EX", "EXNEX", "DPMEX", "DMPP", "UIP",
                       "PBM-HS", "PBM-DL", "PBM-SS", "PBM-SSL")))
```

3.4 Drawing drawing posterior distributions of β_{H_k} for Case 1

```r
fig_HS <- beta_fig(fit_HS, "PBM-HS")
fig_DL <- beta_fig(fit_DL, "PBM-DL")
fig_SS <- beta_fig(fit_SS, "PBM-SS")
fig_SSL <- beta_fig(fit_SSL, "PBM-SSL")

fig_case1 <- ggarrange(fig_HS, fig_DL, fig_SS, fig_SSL,
                       nrow = 4, common.legend = TRUE, legend = "bottom")

print(fig_case1)
```


PBM−HS
PBM−DL
PBM−SS
PBM−SSL
Density
β
−2
0
2
4
0
1
2
β1
β2
β3
β4
β5
β6
β7
β8

```r
ggsave("Beta_Case1.pdf", fig_case1, width = 9, height = 12 , bg = "white")
```

4 Case study 2 (artificial data)

4.1 Datasets

```r
n_h <- c(107, 44, 51, 39, 139, 20, 78, 35)
x_h <- c( 23, 12, 31,  9,  39,  6,  9, 10)
n_CC <- 6
x_CC <- 1
n_CT <- 23
x_CT <- 14

Historical <- data.frame(
  N = n_h,
  X = x_h
)

Current <- data.frame(
  TRT = c(0,1),
  N = c(n_CC, n_CT),
  X = c(x_CC, x_CT)
)

Data_all <- list(Historical = Historical, Current = Current)
ANA <- list(H = length(Data_all$Historical[,"N"]),
            x_h = Data_all$Historical[,"X"], n_h = Data_all$Historical[,"N"],
            x_CC = Data_all$Current$X[1], n_CC = Data_all$Current$N[1],
            x_CT = Data_all$Current$X[2], n_CT = Data_all$Current$N[2],
            betascale = 1, nu = 1, C = 10)
```

4.2 MCMC fitting

```r
fit_Current  <- Current_stan$sample(data = ANA, seed = 999, chains = 4,
        parallel_chains = 4, iter_warmup = 4000, iter_sampling = 4000, refresh = 0)

fit_Pooled <- Pooled_stan$sample(data = ANA, seed = 999, chains = 4,
        parallel_chains = 4, iter_warmup = 4000, iter_sampling = 4000, refresh = 0)

fit_EX     <- EX_stan$sample(data = ANA, seed = 999, chains = 4,
        parallel_chains = 4, iter_warmup = 4000, iter_sampling = 4000,
        adapt_delta = 0.9, refresh = 0)

fit_EXNEX  <- EXNEX_stan$sample(data = ANA, seed = 999, chains = 4,
        parallel_chains = 4, iter_warmup = 4000, iter_sampling = 4000,
        adapt_delta = 0.9, refresh = 0)

fit_DPMEX  <- DPMEX_stan$sample(data = ANA, seed = 999, chains = 4,
```

```
        parallel_chains = 4, iter_warmup = 4000, iter_sampling = 6000,
        adapt_delta = 0.95, refresh = 0)

fit_DMPP   <- DMPP_stan$sample(data = ANA, seed = 999, chains = 4,
        parallel_chains = 4, iter_warmup = 4000, iter_sampling = 6000,
        adapt_delta = 0.95, refresh = 0)

gamma_h = rep(NA, length(Data_all$Historical[,"N"]))
for(h in 1:length(Data_all$Historical[,"N"])){
  gamma_h[h] = min(1, Data_all$Historical[,"N"][h] / Data_all$Current$N[1])
}

theta_h <- rep(NA, length(n_h))
I_U <- rep(NA, length(n_h))
for(h in 1:length(n_h)){
  theta_h[h] = x_h[h] * 1.0 / n_h[h];
  I_U[h] = 1 / (theta_h[h] * (1 - theta_h[h]));
}

ANA_UIP <- list(H = length(Data_all$Historical[,"N"]),
                x_h = Data_all$Historical$X, n_h = Data_all$Historical$N,
                x_CC = Data_all$Current$X[1], n_CC = Data_all$Current$N[1],
                x_CT = Data_all$Current$X[2], n_CT = Data_all$Current$N[2],
                gamma_h = gamma_h, theta_h = theta_h, I_U = I_U)

fit_UIPDir  <- UIPDir_stan$sample(data = ANA_UIP, seed = 999, chains = 4,
        parallel_chains = 4, iter_warmup = 4000, iter_sampling = 8000,
        adapt_delta = 0.999, max_treedepth = 12, refresh = 0)


fit_HS     <- HS_stan$sample(data = ANA, seed = 999, chains = 4,
        parallel_chains = 4, iter_warmup = 4000, iter_sampling = 12000,
        adapt_delta = 0.999, refresh = 0)

ANA_DL <- list(H = length(Data_all$Historical[,"N"]),
               x_h = Data_all$Historical[,"X"], n_h = Data_all$Historical[,"N"],
               x_CC = Data_all$Current$X[1], n_CC = Data_all$Current$N[1],
               x_CT = Data_all$Current$X[2], n_CT = Data_all$Current$N[2], a = 0.5)
fit_DL <- DL_stan$sample(data = ANA_DL, seed = 999, chains = 4,
        parallel_chains = 4, iter_warmup = 4000, iter_sampling = 4000,
        adapt_delta = 0.9, refresh = 0)

fit_SS     <- SS_stan$sample(data = ANA, seed = 999, chains = 4,
        parallel_chains = 4, iter_warmup = 4000, iter_sampling = 6000,
        adapt_delta = 0.95, max_treedepth = 12, refresh = 0)

ANA_SSL <- list(H = length(Data_all$Historical[,"N"]),
                x_h = Data_all$Historical[,"X"], n_h = Data_all$Historical[,"N"],
                x_CC = Data_all$Current$X[1], n_CC = Data_all$Current$N[1],
                x_CT = Data_all$Current$X[2], n_CT = Data_all$Current$N[2],
```

```
                lambda_spike = 0.1, lambda_slab = 2.0)
fit_SSL <- SSL_stan$sample(data = ANA_SSL, seed = 999, chains = 4,
        parallel_chains = 4, iter_warmup = 4000, iter_sampling = 4000,
        adapt_delta = 0.9, refresh = 0)
```

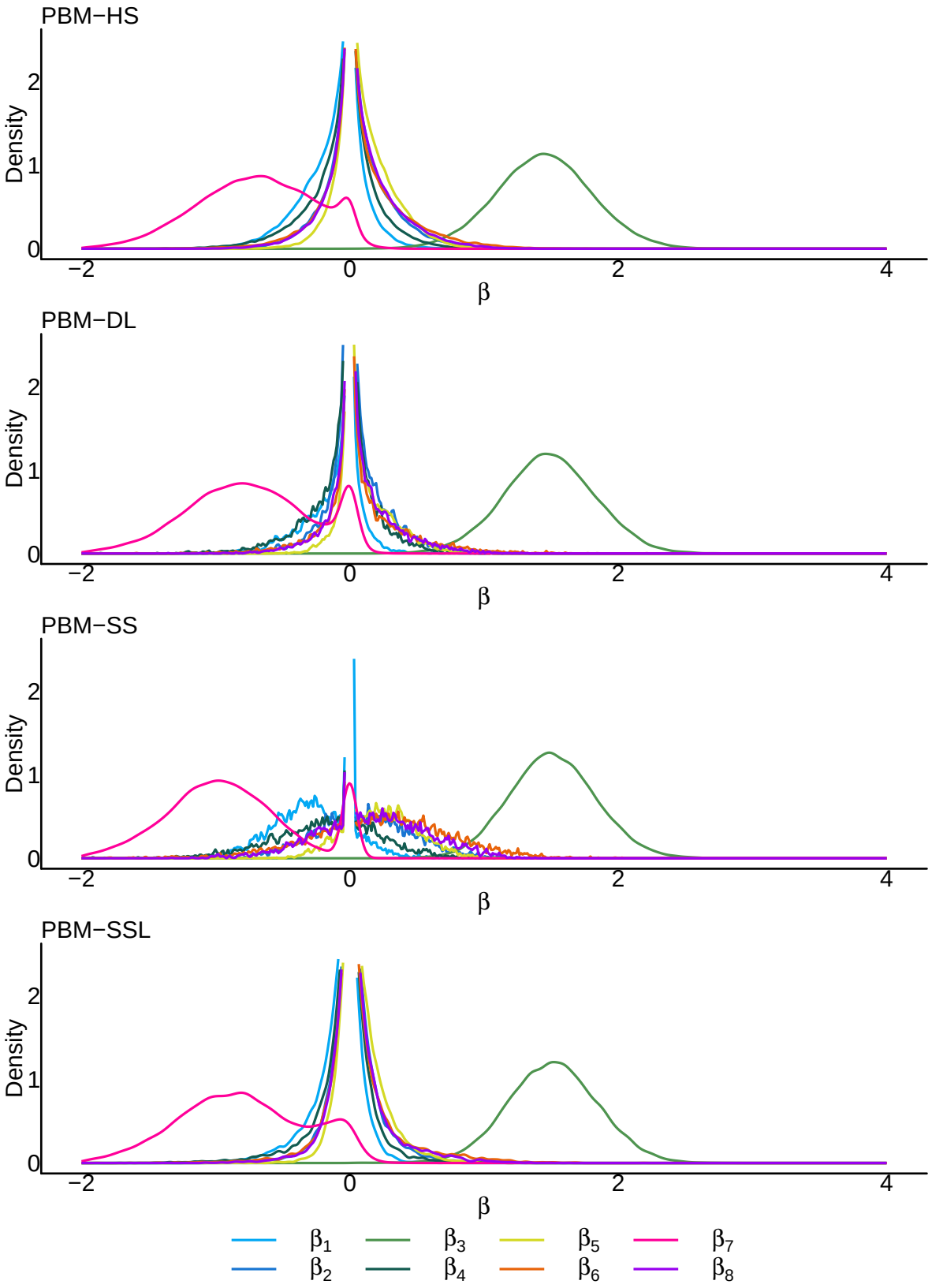
4.3 Aggregate posterior distribution

```
Result_ALL <- matrix(0, nrow = 11, ncol = 7)
Result_ALL[1,]  <- c(1, mean(fit_Current$draws("g_pi")), sd(fit_Current$draws("g_pi")),
                     quantile(fit_Current$draws("g_pi"), 0.025, names = F),
                     quantile(fit_Current$draws("g_pi"), 0.975, names = F),
                     max(fit_Current$summary()$rhat), NA)
Result_ALL[2,]  <- c(2, mean(fit_Pooled$draws("g_pi")), sd(fit_Pooled$draws("g_pi")),
                     quantile(fit_Pooled$draws("g_pi"), 0.025, names = F),
                     quantile(fit_Pooled$draws("g_pi"), 0.975, names = F),
                     max(fit_Pooled$summary()$rhat), NA)
Result_ALL[3,]  <- c(3, mean(fit_EX$draws("g_pi")), sd(fit_EX$draws("g_pi")),
                     quantile(fit_EX$draws("g_pi"), 0.025, names = F),
                     quantile(fit_EX$draws("g_pi"), 0.975, names = F),
                     max(fit_EX$summary()$rhat), ess_calc3(fit_EX))
Result_ALL[4,]  <- c(4, mean(fit_EXNEX$draws("g_pi")), sd(fit_EXNEX$draws("g_pi")),
                     quantile(fit_EXNEX$draws("g_pi"), 0.025, names = F),
                     quantile(fit_EXNEX$draws("g_pi"), 0.975, names = F),
                     max(fit_EXNEX$summary()$rhat), ess_calc3(fit_EXNEX))
Result_ALL[5,]  <- c(5, mean(fit_DPMEX$draws("g_pi")), sd(fit_DPMEX$draws("g_pi")),
                     quantile(fit_DPMEX$draws("g_pi"), 0.025, names = F),
                     quantile(fit_DPMEX$draws("g_pi"), 0.975, names = F),
                     max(fit_DPMEX$summary()$rhat), ess_calc3(fit_DPMEX))
Result_ALL[6,]  <- c(6, mean(fit_DMPP$draws("g_pi")), sd(fit_DMPP$draws("g_pi")),
                     quantile(fit_DMPP$draws("g_pi"), 0.025, names = F),
                     quantile(fit_DMPP$draws("g_pi"), 0.975, names = F),
                     max(fit_DMPP$summary()$rhat), ess_calc3(fit_DMPP))
Result_ALL[7,]  <- c(7, mean(fit_UIPDir$draws("g_pi")), sd(fit_UIPDir$draws("g_pi")),
                     quantile(fit_UIPDir$draws("g_pi"), 0.025, names = F),
                     quantile(fit_UIPDir$draws("g_pi"), 0.975, names = F),
                     max(fit_UIPDir$summary()$rhat), ess_calc3(fit_UIPDir))
Result_ALL[8,]  <- c(8, mean(fit_HS$draws("g_pi")), sd(fit_HS$draws("g_pi")),
                     quantile(fit_HS$draws("g_pi"), 0.025, names = F),
                     quantile(fit_HS$draws("g_pi"), 0.975, names = F),
                     max(fit_HS$summary()$rhat) , ess_calc3(fit_HS))
Result_ALL[9,]  <- c(9, mean(fit_DL$draws("g_pi")), sd(fit_DL$draws("g_pi")),
                     quantile(fit_DL$draws("g_pi"), 0.025, names = F),
                     quantile(fit_DL$draws("g_pi"), 0.975, names = F),
                     max(fit_DL$summary()$rhat), ess_calc3(fit_DL))
Result_ALL[10,]  <- c(10, mean(fit_SS$draws("g_pi")), sd(fit_SS$draws("g_pi")),
                      quantile(fit_SS$draws("g_pi"), 0.025, names = F),
                      quantile(fit_SS$draws("g_pi"), 0.975, names = F),
                      max(fit_SS$summary()$rhat), ess_calc3(fit_SS))
Result_ALL[11,]  <- c(11, mean(fit_SSL$draws("g_pi")), sd(fit_SSL$draws("g_pi")),
```

```
                      quantile(fit_SSL$draws("g_pi"), 0.025, names = F),
                      quantile(fit_SSL$draws("g_pi"), 0.975, names = F),
                      max(fit_SSL$summary()$rhat), ess_calc3(fit_SSL))

Result_ALL2 <- as.data.frame(Result_ALL,stringsAsFactors = F)
colnames(Result_ALL2) <- c("MethodN", "Mean", "SD", "LCI", "UCI", "diag", "ESS")
Result_ALL2 <- Result_ALL2 %>% mutate(Range = (UCI - LCI)) %>%
  mutate(Method = case_when(
    MethodN == 1 ~ "CD",
    MethodN == 2 ~ "PD",
    MethodN == 3 ~ "EX",
    MethodN == 4 ~ "EXNEX",
    MethodN == 5 ~ "DPMEX",
    MethodN == 6 ~ "DMPP",
    MethodN == 7 ~ "UIP",
    MethodN == 8 ~ "PBM-HS",
    MethodN == 9 ~ "PBM-DL",
    MethodN == 10 ~ "PBM-SS",
    MethodN == 11 ~ "PBM-SSL",
    TRUE ~ ""
  )) %>%
  mutate(Mean1 = Mean * 100, SD1 = SD * 100,
         LCI1 = LCI * 100, UCI1 = UCI * 100,
         CI1 = paste0("(", format(round(LCI * 100, digits = 1), nsmall = 1), ", ",
                      format(round(UCI * 100, digits = 1), nsmall = 1), ")")) %>%
  mutate(EHSS1 = if_else(MethodN %in% c(1, 2), "",
                         format(round(ESS - n_CC, digits = 1), nsmall = 1))) %>%
  select(c(Method, Mean1, SD1, LCI1, UCI1, CI1, EHSS1)) %>%
  mutate(case = "2")

Result_ALL2$Method <- factor(Result_ALL2$Method,
        levels = rev(c("CD", "PD", "EX", "EXNEX", "DPMEX", "DMPP", "UIP",
                       "PBM-HS", "PBM-DL", "PBM-SS", "PBM-SSL")))
```

4.4 Drawing drawing posterior distributions of β_{H_k} for Case 2

```
fig_HS <- beta_fig(fit_HS, "PBM-HS")
fig_DL <- beta_fig(fit_DL, "PBM-DL")
fig_SS <- beta_fig(fit_SS, "PBM-SS")
fig_SSL <- beta_fig(fit_SSL, "PBM-SSL")

fig_case2 <- ggarrange(fig_HS, fig_DL, fig_SS, fig_SSL, nrow = 4,
                       common.legend = TRUE, legend = "bottom")

print(fig_case2)
```


PBM−HS
PBM−DL
PBM−SS
PBM−SSL
Density
β
−2
0
2
4
0
1
2
β1
β2
β3
β4
β5
β6
β7
β8

```
ggsave("Beta_Case2.pdf", fig_case2, width = 9, height = 12 ,bg = "white")
```

5 Create output

```
Res_ALL <- rbind(Result_ALL1, Result_ALL2)

Res_ALL_out <- Res_ALL %>%
  select(-c(LCI1, UCI1)) %>%
  pivot_wider(names_from = case, values_from = c(Mean1, SD1, CI1, EHSS1),
              names_sep = "_") %>%
  select(Method, Mean1_1, SD1_1, CI1_1, EHSS1_1, Mean1_2, SD1_2, CI1_2, EHSS1_2)
write.csv(Res_ALL_out, file = "Result.csv", row.names = FALSE)

kable(Res_ALL_out, align = c("lrrcrrrcr"), digits = 1,
      format = "latex", escape = F,
      caption = "Summary of posterior distributions of the treatment effect, $\\pi_{\\rm CT}-\\pi_{\\
      label = "tab1",
      col.names = c("Method", "Mean", "SD", "95\\%CI", "EHSS",
                    "Mean", "SD", "95\\%CI", "EHSS"),
      row.names = FALSE,
      booktabs = T, linesep = "") %>%
      kable_styling(font_size = 11, latex_options = c("hold_position")) %>%
      add_footnote("SD: standard deviation; CI: credible interval; CD: current data analysis; PD: poo
      notation = "none", threeparttable = T) %>%
      add_header_above(c(" ", "Case 1" = 4, "Case 2" = 4))
```

5.1 Drawing forest plot

```
labeli <- as_labeller(c("1" = "Case 1", "2" = "Case 2"))

Res_ALL_fig <- Res_ALL
Res_ALL_fig[Res_ALL_fig$Method == "PD", "EHSS1"] <- "EHSS"

Result_fig <- ggplot(Res_ALL_fig, aes(x = Mean1, y = Method)) +
  xlab("Treatment effect in terms of the response rate (%)") +
  geom_point(size = 3) +
  geom_linerange(aes(xmin = LCI1, xmax = UCI1), size = 1) +
  coord_cartesian(xlim = c(0, 80), expand = T) +
  geom_text(aes(x = 70, label = EHSS1), hjust = 0, size = 4) +
  theme(
    plot.title = element_blank(),
    axis.text.x = element_text(colour = "black", size = 10),
    axis.text.y = element_text(colour = "black", size = 10),
    axis.ticks.x = element_line(colour = "black", size = 1),
    axis.ticks.y = element_line(colour = "black", size = 1),
    axis.title.x = element_text(colour = "black", size = 16),
    axis.title.y = element_blank(),
    axis.line.x = element_line(colour = "#000000", size = 1,
```

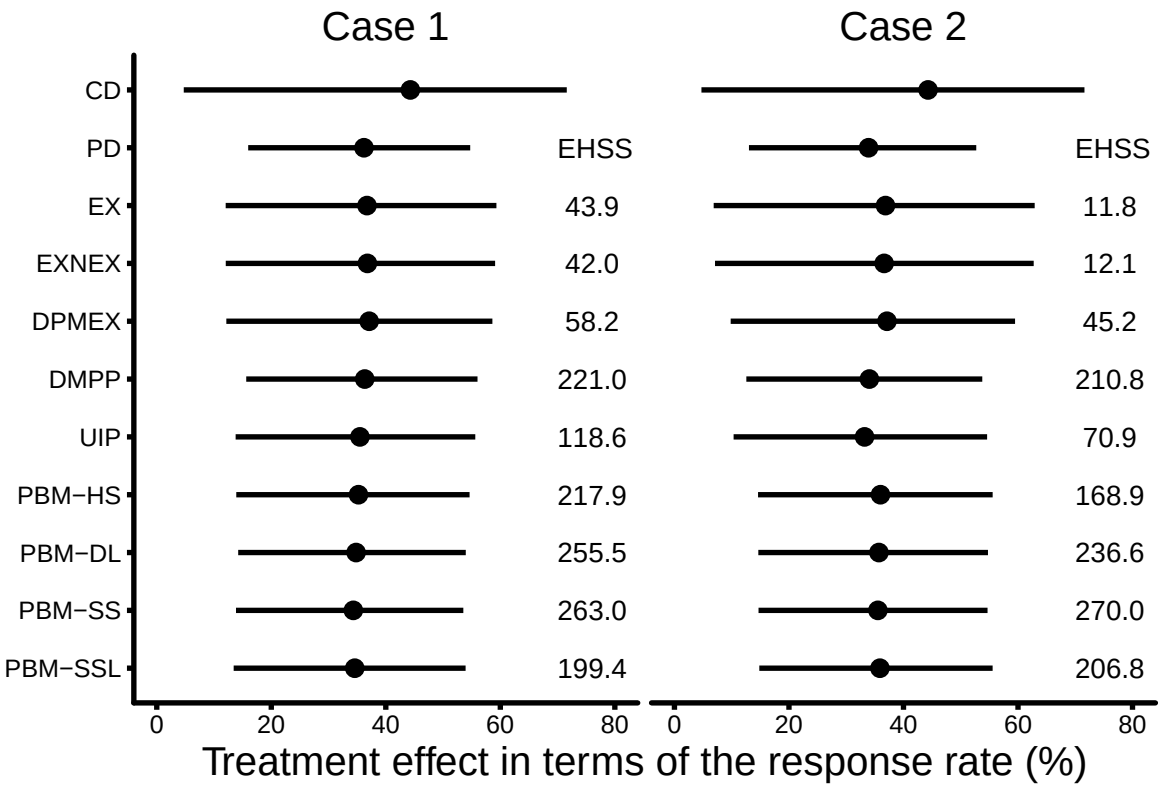
Table 1: Summary of posterior distributions of the treatment effect, $\pi_{CT} - \pi_{CC}(\%)$, for cases 1 (original dataset) and 2 (artificial dataset).

	Case 1				Case 2			
Method	Mean	SD	95%CI	EHSS	Mean	SD	95%CI	EHSS
CD	44.3	17.1	(4.7, 71.6)		44.3	17.1	(4.7, 71.6)	
PD	36.2	10.0	(16.0, 54.7)		33.9	10.1	(13.0, 52.7)	
EX	36.7	12.1	(12.0, 59.3)	43.9	36.9	14.3	(6.9, 62.9)	11.8
EXNEX	36.8	12.0	(12.1, 59.1)	42.0	36.6	14.2	(7.1, 62.7)	12.1
DPMEX	37.1	11.8	(12.2, 58.6)	58.2	37.1	12.7	(9.8, 59.5)	45.2
DMPP	36.3	10.4	(15.6, 56.0)	221.0	34.0	10.6	(12.6, 53.8)	210.8
UIP	35.5	10.8	(13.8, 55.6)	118.6	33.2	11.3	(10.3, 54.6)	70.9
PBM-HS	35.2	10.5	(13.9, 54.6)	217.9	36.0	10.5	(14.6, 55.6)	168.9
PBM-DL	34.8	10.3	(14.2, 54.0)	255.5	35.7	10.2	(14.7, 54.8)	236.6
PBM-SS	34.3	10.3	(13.8, 53.5)	263.0	35.5	10.3	(14.7, 54.7)	270.0
PBM-SSL	34.6	10.4	(13.4, 53.9)	199.4	35.9	10.5	(14.8, 55.6)	206.8

SD: standard deviation; CI: credible interval; CD: current data analysis; PD: pooled data analysis; EX: full exchangeability MAC method; EXNEX: exchangeability–non-exchangeability MAC method; DPMEX: Dirichlet process mixture of exchangeability MAC method; DMPP: dependent modified power prior; UIP: unit information prior; PBM-HS: potential bias model with horseshoe prior; PBM-DL: potential bias model with Dirichlet–Laplace prior; PBM-SS: potential bias model with spike-and-slab prior; PBM-SSL: potential bias model with spike-and-slab Lasso prior.

```
                          linetype = "solid", lineend = "round"),
    axis.line.y = element_line(colour = "#000000", size = 1,
                          linetype = "solid", lineend = "round"),
    panel.background = element_rect(fill = "transparent", colour = NA),
    panel.border = element_blank(),
    panel.grid = element_blank(),
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    panel.grid.major.x = element_blank(),
    panel.grid.major.y = element_blank(),
    panel.grid.minor.x = element_blank(),
    panel.grid.minor.y = element_blank()
  ) +
  facet_grid( ~ case, scales = "fixed", space = "free",
              labeller = labeli, switch = "y") +
  theme(
    strip.placement.y = "outside",
    strip.background = element_blank(),
    strip.text.x = element_text(colour = "black", size = 16),
    panel.spacing.y = unit(1.5, "lines")
  )
```

```
print(Result_fig)
```




```
ggsave("Result.pdf", Result_fig, width = 12, height = 8, bg = "white")
```