```
######### Declarations #################
# Author: Zichen Xu
# Affiliation: Beijing Normal University(BNU) and National Center for
Cardiovascular Diseases(NCCD)
# Email Address: zichenxu407@gmail.com
# Final edition: Dec/2023

######### Loading data and packages #############

# Preparation
# install.packages("vcdExtra")
# install.packages("tidyr")
# install.packages("lme4")
# install.packages("lmerTest")
# install.packages("readr")
# install.packages("mvmeta")
# install.packages("BRugs")
# install.packages("BayesPPD")
# install.packages("DescTools")
# install.packages("rstan")
# install.packages("mgcv")
# install.packages("arm")
# install.packages("magrittr")
# install.packages("MASS")
# install.packages("remotes")
# library("remotes")
# remotes::install_github("maxbiostat/npowerPrioR")
# install.packages("parallel")
# library("parallel")
# install.packages("future")
# library("future")
# install.packages("future.apply")
# library("future.apply")
library("vcdExtra")
library("stats")
library("tidyr")
library("lme4")
library("lmerTest")
library("readr")
library("mvmeta")
library("BRugs")
library("BayesPPD")
library("DescTools")
library("rstan")
library("mgcv")
library("arm")
library("magrittr")
library("MASS")
library("npowerPrioR")
library("parallel")
library("future")
library("future.apply")

# Import the data
IIa_fas <- read_csv("C:/Users/Think/Desktop/NCCD/11 My_Combination_of_ab/
IIa_fas.csv")
IIa_pps <- read_csv("C:/Users/Think/Desktop/NCCD/11 My_Combination_of_ab/
IIa_pps.csv")
IIb_fas <- read_csv("C:/Users/Think/Desktop/NCCD/11 My_Combination_of_ab/
IIb_fas.csv")
```

```
IIb_pps <- read_csv("C:/Users/Think/Desktop/NCCD/11 My_Combination_of_ab/
IIb_pps.csv")

######### Preprocessing Data ######################

# Descriptions of FAS_IIa
names(IIa_fas)
names(IIa_fas) <- c("Center", "ID", "Sex", "Height", "Smoke", "JL",
"Group", "Timi", "Age")
names(IIa_pps) <- c("Center", "ID", "Sex", "Height", "Smoke", "JL",
"Group", "Timi", "Age")
names(IIb_fas) <- c("Center", "ID", "Group", "Timi", "Sex", "Height",
"Smoke", "JL", "Age")
names(IIb_pps) <- c("Center", "ID", "Group", "Timi", "Sex", "Height",
"Smoke", "JL", "Age")

# Modify the label"3" of Group in IIb
IIb_fas$Group[IIb_fas$Group == 3] <- 4
IIb_pps$Group[IIb_pps$Group == 3] <- 4

# Missing value control
attach(IIb_fas)
#sum(is.na(IIb_fas$Timi)) # 4 missing values
# Note that there is a missing value in Height

# dealing with the missing values
# remove directly(Complete Case Analysis)
IIb_fas_cca <- IIb_fas %>% drop_na(Timi)
#sum(is.na(IIb_fas_cca$Timi)) # no missing value already

# WOCF(Worst Observations Carry Forward)
IIb_fas_WOCF <- IIb_fas
IIb_fas_WOCF$Timi[is.na(IIb_fas_WOCF$Timi)] <- 0
#sum(is.na(IIb_fas_WOCF$Timi)) # no missing value already
detach(IIb_fas)

# Preprocess the data and create a binary variable
preprocess_data <- function(data) {
  # Check for missing values
  missing_values <- sum(is.na(data$Timi))
  cat("Number of missing values:", missing_values, "\n")

  # Create a binary variable Timi23
  data$Timi23 <- ifelse(data$Timi >= 2, 1, 0)

  # Display the table of Timi23 values
  cat("The frequency for each category is:")
  table(data$Timi23)
  return(data)
}

IIa_fas <- preprocess_data(IIa_fas)
IIa_pps <- preprocess_data(IIa_pps)
IIb_fas_cca <- preprocess_data(IIb_fas_cca)
IIb_pps <- preprocess_data(IIb_pps)
IIb_fas_WOCF <- preprocess_data(IIb_fas_WOCF)

# Combining directly for FAS(only cca) and pps
ab_fas_dir_cca <- rbind(IIa_fas, IIb_fas_cca)
ab_pps_dir <- rbind(IIa_pps, IIb_pps)
ab_fas_dir_wocf <- rbind(IIa_fas, IIb_fas_WOCF)
```

```
# Add a label to identify historical and current data
# 1 for Historical data; 2 for current data.
trialid <- c(rep(1, nrow(IIa_fas)), rep(2, nrow(IIb_fas_cca)))
ab_fas_dir_cca <- cbind(ab_fas_dir_cca, trialid)

trialid_w <- c(rep(1, nrow(IIa_fas)), rep(2, nrow(IIb_fas_WOCF)))
ab_fas_dir_wocf <- cbind(ab_fas_dir_wocf, trialid_w)

# Encode and pair treatment+control separately(For IPD and BHM)
# Standardize and subgroup the dataset to simplify "Group" into a binary
variable.
attach(ab_fas_dir_cca)
ab_fas_14 <- subset(ab_fas_dir_cca, Group == 1|Group == 4)
ab_fas_14$Group[ab_fas_14$Group == 1] <- 1 # treatment
ab_fas_14$Group[ab_fas_14$Group == 4] <- 0

ab_fas_24 <- subset(ab_fas_dir_cca, Group == 2|Group == 4)
ab_fas_24$Group[ab_fas_24$Group == 2] <- 1 # treatment
ab_fas_24$Group[ab_fas_24$Group == 4] <- 0
detach(ab_fas_dir_cca)

######### Statistical Description ####################

# Description Statistical Analysis
# Repeat the protocol.

# Analyze and display the contingency table originally
analyze_contingency_table <- function(data) {
  tbl <- table(data$Timi, data$Group)
  print(tbl)
  prop_table <- prop.table(tbl1, 2)
  print(prop_table)
}

# Analyze and display the contingency table under Timi23
analyze_contingency_table23 <- function(data) {
  tbl <- table(data$Timi23, data$Group)
  print(tbl)
  prop_table <- prop.table(tbl1, 2)
  print(prop_table)
}



analyze_contingency_table(IIa_fas)
#   1 2 3 4 5
# 0 3 2 2 3 1
# 2 3 1 1 2 2
# 3 5 8 5 5 8


analyze_contingency_table(IIa_pps)
#   1 2 3 4 5
# 0 2 1 2 2 1
# 2 3 1 0 1 2
# 3 5 7 4 5 7

analyze_contingency_table23(IIb_fas_cca)
#    1  2  4
```

```
# 0 40 32 46
# 1 73 82 68

analyze_contingency_table23(IIb_pps)
#    1  2  4
# 0 38 29 45
# 1 72 80 64

# Results are exactly the same as our original analyses.

analyze_contingency_table23(ab_fas_dir_cca)
#    1  2  3  4  5
# 0 43 34  2 49  1
# 1 81 91  6 75 10


analyze_contingency_table23(ab_pps_dir)
#    1  2  3  4  5
# 0 40 30  2 47  1
# 1 80 88  4 70  9

######### Repeat the Protocol: Chi-square ##############


# Analysis
# Repeat the protocol

# Subset control-treatment groups and perform Chi-square test
perform_chi_squared_test <- function(data, trt_grp) {
  # Subset the data based on Group values
  dt <- subset(data, Group %in% c(trt_grp, 4))
  # Create a contingency table
  tb <- xtabs(~ Timi23 + Group, data = dt)
  # Perform a chi-squared test
  chi_rslt <- chisq.test(tb, correct = FALSE)
  return(chi_rslt)
}


perform_chi_squared_test(IIb_fas_cca, trt_grp = 1)
# Note: IIb protocol do not correct the continuity
# X-squared = 0.59152, df = 1,
# p-value = 0.4418
# Exactly the same.

perform_chi_squared_test(IIb_fas_cca, trt_grp = 2)
# X-squared = 3.8195, df = 1,
# p-value = 0.05066
# Almost the same.

perform_chi_squared_test(IIb_pps, trt_grp = 1)
# X-squared = 1.0564, df = 1,
# p-value = 0.304
# Exactly the same.

perform_chi_squared_test(IIb_pps, trt_grp = 2)
# X-squared = 5.2372, df = 1,
# p-value = 0.02211
# Almost the same.

# Calculate the power of Chi-square
```

```
# Original IIb
power.chisq.test(n=227, w=0.2, df=1, sig.level = 0.05, power = NULL)# small
effect
# power = 0.8539077
power.chisq.test(n=228, w=0.2, df=1, sig.level = 0.05, power = NULL)# small
effect
# power = 0.8554211
power.chisq.test(n=227, w=0.5, df=1, sig.level = 0.05, power = NULL)#
moderate effect
# power = 1

# Original IIa
perform_chi_squared_test(IIa_fas, trt_grp = 1) # p-value = 0.8901
perform_chi_squared_test(IIa_fas, trt_grp = 2) # p-value = 0.5254

######### Combining Chi-square ######################

# Analyze the combined dataset
perform_chi_squared_test(ab_fas_dir_cca, trt_grp = 1)
# X-squared = 0.62207, df = 1,
# p-value = 0.4303

perform_chi_squared_test(ab_fas_dir_cca, trt_grp = 2)
# X-squared = 4.2491, df = 1,
# p-value = 0.03927

perform_chi_squared_test(ab_pps_dir, trt_grp = 1)
# X-squared = 1.1921, df = 1,
# p-value = 0.2749

perform_chi_squared_test(ab_pps_dir, trt_grp = 2)
# X-squared = 5.7997, df = 1,
# p-value = 0.01603

power.chisq.test(n=248, w=0.1, df=1, sig.level = 0.05, power = NULL)
# power = 0.3502626
power.chisq.test(n=248, w=0.12, df=1, sig.level = 0.05, power = NULL)
# power = 0.4720755
power.chisq.test(n=248, w=0.15, df=1, sig.level = 0.05, power = NULL)
# power = 0.6562534
power.chisq.test(n=248, w=NULL, df=1, sig.level = 0.05, power = 0.5)
# w = 0.1244504

######### IPD-Meta Two Stage #################


# IPD-Meta Two-stage analysis
## From now on, every analysis is based on fas cca dataset.

# Initialization and preparations
rslt2_1_14 <- as.data.frame(array(NA, dim=c(2,2)))
rslt2_1_24 <- as.data.frame(array(NA, dim=c(2,2)))
# Note there are only one covariate(Group) in our model.
# And, two trials right now.
colnames(rslt2_1_14) <- c("betai", "var_betai")
colnames(rslt2_1_24) <- c("betai", "var_betai")


# Stage I
fit_logistic_model <- function(data, trial_num) {
```

```
  results <- as.data.frame(array(NA, dim=c(trial_num,2)))  # Initialize a
results matrix
  colnames(results) <- c("betai", "var_betai")
  for (i in 1:trial_num) {
    dsi <- as.data.frame(data[which(data$trialid == i), ])
    fit <- glm(Timi23 ~ Group, data = dsi, family = binomial(link =
"logit"))
    results[i, ] <- c(coefficients(fit)[2], vcov(fit)[2, 2])
    # Note that coef is the (2,2) element, because there is an intercept.
  }
  return(results)
}

rslt_IPD_14 <- fit_logistic_model(data = ab_fas_14, trial_num = 2)
#       betai  var_betai
# 1 0.1335314 0.93452265
# 2 0.2107137 0.07514364

rslt_IPD_24 <- fit_logistic_model(data = ab_fas_24, trial_num = 2)
#       betai  var_betai
# 1 0.6567795 1.08730108
# 2 0.5501170 0.07989007

# Stage II
fit_mixed_effects_model <- function(results) {
  model <- mvmeta(betai ~ 1, S = var_betai, data = results, method =
"reml")
  return(model)
}

fit2_2_14 <- fit_mixed_effects_model(rslt_IPD_14)
summary(fit2_2_14)
## coef: 0.2050(SE:0.2637) | between-study heterogeneity(tau_beta): 0.0001
## I-square statistic = 1.0%
## It is insignificant(p-value: 0.4370)
confint(fit2_2_14)
#                 2.5 %    97.5 %
# (Intercept) -0.311923 0.7218619

coef(fit2_2_14)# 0.2049694
sqrt(vcov(fit2_2_14))# 0.2637255

# Fixed effect model(standard GLS - inverse probability weighted methods)
fit2_2_14_fix <- mvmeta(betai ~ 1, S = var_betai, data = rslt_IPD_14,
method = "fixed")
summary(fit2_2_14_fix)
# Treatment effect: 2050(SE: 0.2637), between-group heterogeneity was set
to be 0
# It is insignificant(p = 0.4370)
# Calculation manually
# > ((0.1335314/0.93452265)+(0.2107137/0.07514364))/(1/0.93452265 +
1/0.07514364)
# [1] 0.2049695

coef(fit2_2_14_fix)# 0.2049694
sqrt(vcov(fit2_2_14_fix))# 0.2637255

fit2_2_24 <- fit_mixed_effects_model(rslt_IPD_24)
summary(fit2_2_24)
## coef: 0.5574(SE: 0.2728) | between-study heterogeneity: 0.0001
## I-square statistic = 1.0%
```

```
## It seems better and significant(p-value: 0.0410).
confint(fit2_2_24)
#                  2.5 %   97.5 %
# (Intercept) 0.02273217 1.092103
coef(fit2_2_24)# 0.5574177
sqrt(vcov(fit2_2_24))# 0.2728038

# Fixed effect model
fit2_2_24_fix <- mvmeta(betai ~ 1, S = var_betai, data = rslt_IPD_24,
method = "fixed")
summary(fit2_2_24_fix)
# Treatment effect: 0.5574(SE: 0.2728)
# It is significant(p = 0.0410)
coef(fit2_2_24_fix)# 0.5574177
sqrt(vcov(fit2_2_24_fix))# 0.2728037

twostage_lg_fit <- function(data, trial_num){
  result_1st <- fit_logistic_model(data, trial_num)
  result_2nd <- fit_mixed_effects_model(result_1st)
  return(result_2nd)
}

######### IPD-Meta One Stage #################


# IPD-Meta One-stage Analysis
IPD14 <- glmer(Timi23 ~ trialid + Group + (Group+0|trialid), family =
binomial, data = ab_fas_14)
summary(IPD14)
confint(IPD14)
IPD14try <- glmer(Timi23 ~ trialid + Group + (Group|trialid), family =
binomial, data = ab_fas_14)
summary(IPD14try)
confint(IPD14try)

# Trialid take the place of intercept.
# It is the baseline of our regression, the study effect.
## Treatment effect: 0.2050(SE: 0.2637) | between-study heterogeneity: 0
## It is insignificant(p = 0.437).

#                   2.5 %    97.5 %
# .sig01       0.0000000 0.8297065
# (Intercept) -0.6164542 3.3232640
# trialid     -1.4805968 0.5246239
# Group       -0.4018611 0.8142913

fixef(IPD14)["Group"]# 0.2049719
se.fixef(IPD14)["Group"]# 0.263712

IPD24 <- glmer(Timi23 ~ trialid + Group + (Group+0|trialid), family =
binomial, data=ab_fas_24)

summary(IPD24)
confint(IPD24)
## Treatment effect: 0.5575(SE: 0.2728) | between-study heterogeneity: 0
## It is significant(p = 0.041).

#                   2.5 %    97.5 %
# .sig01       0.00000000 0.9065761
# (Intercept) -0.53559165 3.6661617
# trialid     -1.65686785 0.4831038
```

```
# Group       -0.06600734 1.2414066

fixef(IPD24)["Group"]# 0.5574595
se.fixef(IPD24)["Group"]# 0.2727825

# Laplace approximation (Default: nAGQ = 1, so one-stage meta is the GLMM
with laplace approximation)
IPD_LA_14 <- glmer(Timi23 ~ trialid + Group + (Group+0|trialid), family =
binomial, data = ab_fas_14, nAGQ = 1)
summary(IPD_LA_14)
# Treatment effect: 0.2050(SE: 0.2637)
# It is insignificant also.(p = 0.437)
confint(IPD_LA_14)
#                  2.5 %    97.5 %
# .sig01       0.0000000 0.8297065
# (Intercept) -0.6164542 3.3232640
# trialid     -1.4805968 0.5246239
# Group       -0.4018611 0.8142913

fixef(IPD_LA_14)["Group"]# 0.2049719
se.fixef(IPD_LA_14)["Group"]# 0.263712
confint(IPD_LA_14)

IPD_LA_24 <- glmer(Timi23 ~ trialid + Group + (Group+0|trialid), family =
binomial, data=ab_fas_24, nAGQ = 1)
# IPD_LA_24_111 <- glmer(Timi23 ~ trialid + Group + (Group|trialid), family
= binomial, data=ab_fas_24, nAGQ = 1)
# summary(IPD_LA_24_111)
summary(IPD_LA_24)
# Treatment effect: 0.5575(SE: 0.2728)
# It is significant(0.041)
confint(IPD_LA_24)
#                   2.5 %    97.5 %
# .sig01       0.00000000 0.9065761
# (Intercept) -0.53559165 3.6661617
# trialid     -1.65686785 0.4831038
# Group       -0.06600734 1.2414066

fixef(IPD_LA_24)["Group"]# 0.5574595
se.fixef(IPD_LA_24)["Group"]# 0.2727825

# PQL
IPD_PQL_14 <- glmmPQL(Timi23 ~ trialid + Group, random = ~Group+0|trialid,
                      family = binomial, data = ab_fas_14, verbose = FALSE,
niter = 10)
summary(IPD_PQL_14)
# Treatment effect: 0.2049718(SE: 0.2653271)
# It is insignificant(p = 0.4405)
sqrt(vcov(IPD_PQL_14)) # 0.26371739 Why are they different?
intervals(IPD_PQL_14)


IPD_PQL_24 <- glmmPQL(Timi23 ~ trialid + Group, random = ~Group+0|trialid,
                      family = binomial, data = ab_fas_24, verbose = FALSE)
summary(IPD_PQL_24)
# Treatment effect: 0.5574595(SE: 0.2744102)
# It is significant(p = 0.0433)
sqrt(vcov(IPD_PQL_24)) # 0.2727521
intervals(IPD_PQL_24, which = "fixed")

######### GLM for Original Dataset ##############
```

```
# Prepare the dataset
rslt_vir_14 <- as.data.frame(array(NA, dim=c(1,2)))
rslt_vir_24 <- as.data.frame(array(NA, dim=c(1,2)))
colnames(rslt_vir_14) <- c("betai", "var_betai")
colnames(rslt_vir_24) <- c("betai", "var_betai")

attach(IIb_fas_cca)
IIb_fas_vir_14 <- subset(IIb_fas_cca, Group == 1|Group == 4)
IIb_fas_vir_14$Group[IIb_fas_vir_14$Group == 1] <- 1 # treatment
IIb_fas_vir_14$Group[IIb_fas_vir_14$Group == 4] <- 0

IIb_fas_vir_24 <- subset(IIb_fas_cca, Group == 2|Group == 4)
IIb_fas_vir_24$Group[IIb_fas_vir_24$Group == 2] <- 1 # treatment
IIb_fas_vir_24$Group[IIb_fas_vir_24$Group == 4] <- 0
detach(IIb_fas_cca)

# Two-stage Meta
fit_vir_14 <- glm(Timi23 ~ Group, data=IIb_fas_vir_14, family =
binomial(link = "logit")) # Fit the model

fit_vir_24 <- glm(Timi23 ~ Group, data=IIb_fas_vir_24, family =
binomial(link = "logit")) # Fit the model

summary(fit_vir_14)
## Treatment Effect: 0.2107(SE: 0.2741)
## It is insignificant(p = 0.4421).

confint(fit_vir_14)
#                   2.5 %    97.5 %
# (Intercept)  0.02011276 0.7706708
# Group       -0.32596817 0.7504989


summary(fit_vir_24)
## Treatment Effect: 0.5501(SE: 0.2826)
## This is much better(p = 0.0516).
## The strength of "Combining" might be hidden due to the imbalance between
two trials.
confint(fit_vir_24)
#                     2.5 %    97.5 %
# (Intercept)  0.0201127591 0.7706708
# Group       -0.0004550502 1.1099176

# IIa result
IIa_glm_24 <-
glm(ab_fas_24$Timi23[ab_fas_24$trialid==1]~ab_fas_24$Group[ab_fas_24$trialid==1],
family = "binomial")
summary(IIa_glm_24)
confint(IIa_glm_24)
# Treatment effect: 0.6568(SE: 1.0427), p-value = 0.529
#                                             2.5 %   97.5 %
# (Intercept)                              -0.4328021 2.382047
# ab_fas_24$Group[ab_fas_24$trialid == 1] -1.3807395 2.886597
IIa_glm_14 <-
glm(ab_fas_14$Timi23[ab_fas_14$trialid==1]~ab_fas_14$Group[ab_fas_14$trialid==1],
family = "binomial")
summary(IIa_glm_14)
confint(IIa_glm_14)
# Treatment effect: 0.1335(SE: 0.9667), p-value = 0.89
#                                             2.5 %   97.5 %
```

```
# (Intercept)                                -0.4328021 2.382047
# ab_fas_14$Group[ab_fas_14$trialid == 1] -1.8157116 2.088450



fit_dir_ab14 <- glm(Timi23 ~ trialid + Group, data=ab_fas_14, family =
binomial(link = "logit")) # Fit the model

fit_dir_ab24 <- glm(Timi23 ~ trialid + Group, data=ab_fas_24, family =
binomial(link = "logit")) # Fit the model

summary(fit_dir_ab14)
## Treatment Effect: 0.2050(SE: 0.2637)
## It is insignificant(p = 0.437).
confint(fit_dir_ab14)
#                  2.5 %    97.5 %
# (Intercept) -0.6163845 3.3235454
# trialid     -1.4807296 0.5245770
# Group       -0.3114115 0.7240712
coef(fit_dir_ab14) # 0.2049719
sqrt(vcov(fit_dir_ab14)) # 0.26371198

summary(fit_dir_ab24)
## Treatment Effect: 0.5575(SE: 0.2728)
## It is significant(p = 0.041).
confint(fit_dir_ab24)
#                   2.5 %    97.5 %
# (Intercept) -0.53572677 3.6664531
# trialid     -1.65702671 0.4830719
# Group        0.02609251 1.0976075
coef(fit_dir_ab24) # 0.5574595
sqrt(vcov(fit_dir_ab24)) # 0.2727825

######### Test-then-pool ##################################################

# Test Statistics
TTP_chi_squared_test <- function(data, grp) {
  # Subset the data based on Group values
  dt <- subset(data, Group==grp)
  # Create a contingency table
  tb <- xtabs(~ Timi23 + trialid, data = dt)
  # Perform a chi-squared test
  chi_rslt <- chisq.test(tb, correct = FALSE)
  return(chi_rslt)
}

# If the response rate between historical and current trial is different,
drop it.
TTP_chi_squared_test(ab_fas_24, grp = 0) # p-value = 0.5209
TTP_chi_squared_test(ab_fas_24, grp = 1) # p-value = 0.4815
# Exchangeable! Now, let's pool them together.

TTP_chi_squared_test(ab_fas_14, grp = 0) # p-value = 0.5209. It's the same.
TTP_chi_squared_test(ab_fas_14, grp = 1) # p-value = 0.5888

######### Bayesian Frameworks ##########################
######### Standard BHM 14 direct pooling
(Binary)#################################

# References: Peter F. Thall et al.(2003)
set.seed(9999)
```

```
# nrep <- 3 # When doing power analysis repeatedly.

# Bayesian Hierarchical Modeling
# For dataset 14

x_bin_14 <- ab_fas_14$Group
y_bin_14 <- ab_fas_14$Timi23

n_14 <- 248

# prior specifications
mean_mu <- 0
prec_mu <- 0.001
mean_mu0 <- 0
prec_mu0 <- 0.001 # for convenience, setting them to be the same

# Pre-specify several borrowing rates.
tau_alpha <- c(0,0,0); tau_beta <- c(0,0,0)
tau_alpha[1] <- 0.01;  tau_beta[1] <- 10000    # lots of borrowing
tau_alpha[2] <- 10;  tau_beta[2] <- 0.1     # a little borrowing
tau_alpha[3] <- 16;  tau_beta[3] <- 16  # not much borrowing

# multi-level for the borrowing
tau_alpha0 <- c(0,0,0); tau_beta0 <- c(0,0,0)
tau_alpha0[1] <- 0.01;  tau_beta0[1] <- 10000    # lots of borrowing
tau_alpha0[2] <- 6;  tau_beta0[2] <- 6     # a little borrowing
tau_alpha0[3] <- 16;  tau_beta0[3] <- 16  # not much borrowing

dput(list(mu=1, prec=0.001, mu0=1, prec0=0.001),"14_bin_inits.txt")
beta1stat_14_bin <- NULL; beta1id <- 0
# for (k in 1:length(tau_alpha0)) {
  k=1;j=1 # for convenience.
  # for (j in 1:length(tau_alpha)){
    bt1 <- NULL; bt0 <- NULL; mu <- NULL; mu0 <- NULL; prec <- NULL; prec0
<- NULL
    dput(list(x=x_bin_14,y=y_bin_14,n=n_14,mean.Mu=mean_mu,prec.Mu=prec_mu,
              mean.Mu0=mean_mu0,prec.Mu0=prec_mu0,
              tau.alpha=as.numeric(tau_alpha[j]),
              tau.beta=as.numeric(tau_beta[j]),
              tau.alpha0=as.numeric(tau_alpha0[k]),
              tau.beta0=as.numeric(tau_beta0[k])),"14_bin_data.txt")

    modelCheck("BUGS_binary.txt")
    modelData("14_bin_data.txt")
    modelCompile(numChains=4)
    modelInits("14_bin_inits.txt")
    modelGenInits()

    modelUpdate(5000)
    samplesSet(c("beta1","beta0","mu","mu0","prec","prec0"))
    modelUpdate(50000) # 50000 is not enough for convergence at the given
precision level.
    # beta1id <- beta1id + 1
    bt1 <- cbind(samplesStats("beta1"), tau_alpha[j], tau_beta[j],
tau_alpha0[k], tau_beta0[k], beta1id)
    bt0 <- cbind(samplesStats("beta0"), "-", "-", "-", "-", "-")
    mu <- cbind(samplesStats("mu"), "-", "-", "-", "-", "-")
    mu0 <- cbind(samplesStats("mu0"),"-", "-", "-", "-", "-")
    prec <- cbind(samplesStats("prec"),"-", "-", "-", "-", "-")
    prec0 <- cbind(samplesStats("prec0"),"-", "-", "-", "-", "-")
```

```
    beta1stat_14_bin <- rbind(beta1stat_14_bin, bt1,bt0,mu,mu0,prec,prec0)

    # some plots
    # samplesHistory("beta1")
    # samplesDensity("beta1") # Posterior distr.
    # beta1store_14_bin <- cbind(beta1store_14_bin, samplesSample("beta1"))
# storing all of the data from the chain.
    # samplesAutoC("beta1", 1)

  cat("\nstats for beta1:\n"); print(beta1stat_14_bin)

  # cat("\n\nHere are simulated results for tau prior no.",j,"and tau0
prior no.",k,
  #     ",\n  and true mean_mu, prec_mu, tau_alpha, tau_beta, tau_alpha0,
tau_beta0 are:
",mean_mu,prec_mu,tau_alpha[j],tau_beta[j],tau_alpha0[k],tau_beta0[k],"\n")
  # }
# }

forest_14_bin <- ggplot(beta1stat_14_bin, aes(mean, ))+
                          geom_point(size=5, color="orange")+
                          geom_errorbarh(aes(xmax=val97.5pc,
xmin=val2.5pc), size=1, height=0.1,color="orange")+
                          scale_x_continuous(limits = c(0.19, 0.22))+
                          geom_vline(aes(xintercept=0), color="gray",
linetype="dashed", size=1.5)+
                          xlab("effect size (beta1)")+
                          ylab(" ")+
                          theme_few()+
                          theme(axis.text.x = element_text(size = 14, color
= "black"))+
                          theme(axis.text.y = element_text(size = 1, color
= "black"))+
                          theme(title = element_text(size = 14))

# beta1stat_14_bin1_20 <- beta1stat_14_bin
# store <- rbind(beta1stat_14_bin1_20, beta1stat_14_bin)
# write.csv(store, file = "sensitivity test for SHM.csv")

# stats for beta1:
#          mean     sd  MC_error val2.5pc median val97.5pc start  sample
tau_alpha[j] tau_beta[j] tau_alpha0[k] tau_beta0[k] beta1id
# beta1  0.2094 0.2633 0.0001467  -0.3060 0.2091    0.7267 50001 5000000
2           2             2            2       1
# beta11 0.2093 0.2635 0.0001415  -0.3064 0.2089    0.7270 50001 5000000
6           6             2            2       2
# beta12 0.2093 0.2634 0.0001400  -0.3065 0.2090    0.7264 50001 5000000
16          16             2            2       3

######### Standard BHM 24 direct pooling
(Binary)#########################################

# Bayesian Hierarchical Modeling
# For dataset 24
set.seed(9999)
x_bin_24 <- ab_fas_24$Group
y_bin_24 <- ab_fas_24$Timi23

n_24 <- 249

beta1store_24_bin <- NULL
```

```
# pwrbeta1_14_bin <- rep(0,length(tau_alpha))

dput(list(mu=1, prec=0.1, mu0=1, prec0=0.1),"24_bin_inits.txt")
beta1stat_24_bin <- NULL; beta1id <- 0
# for (k in 1:length(tau_alpha0)) {
k=1 # for convenience.
# for (j in 1:length(tau_alpha)){
j=1
  bt1_new <- NULL;
  dput(list(x=x_bin_24,y=y_bin_24,n=n_24,mean.Mu=mean_mu,prec.Mu=prec_mu,
            mean.Mu0=mean_mu0,prec.Mu0=prec_mu0,
            tau.alpha=as.numeric(tau_alpha[j]),
            tau.beta=as.numeric(tau_beta[j]),
            tau.alpha0=as.numeric(tau_alpha0[k]),
            tau.beta0=as.numeric(tau_beta0[k])),"24_bin_data.txt")

  modelCheck("BUGS_binary.txt")
  modelData("24_bin_data.txt")
  modelCompile(numChains=1)
  modelInits("24_bin_inits.txt")
  modelGenInits()

  modelUpdate(50000)
  samplesSet(c("beta1"))
  modelUpdate(5000000) # 50000 is not enough for convergence.
  beta1id <- beta1id + 1
  bt1_new <- cbind(samplesStats("beta1"), tau_alpha[j], tau_beta[j],
tau_alpha0[k], tau_beta0[k], beta1id)
  beta1stat_24_bin <- rbind(beta1stat_24_bin, bt1_new)

  # some plots
  # samplesHistory("beta1")
  # samplesDensity("beta1") # Posterior distr.
  beta1store_24_bin <- cbind(beta1store_24_bin, samplesSample("beta1")) #
storing all of the data from the chain.
  # samplesAutoC("beta1", 1)

  cat("\nstats for beta1:\n"); print(beta1stat_24_bin)
# }
# }

# stats for beta1:
#         mean     sd  MC_error val2.5pc median val97.5pc start  sample
tau_alpha[j] tau_beta[j] tau_alpha0[k] tau_beta0[k] beta1id
# beta1 0.5619 0.2725 0.0001575  0.03042 0.5609     1.099 50001 5000000
2           2             2            2       1

######### BHM Real Hierarchical for 14 ########################
set.seed(9999)

# nrep <- 3

# Bayesian Hierarchical Modeling
# For dataset 14

x0_bin_14 <- ab_fas_14$Group[ab_fas_14$trialid == "1"]
y0_bin_14 <- ab_fas_14$Timi23[ab_fas_14$trialid == "1"]
x1_bin_14 <- ab_fas_14$Group[ab_fas_14$trialid == "2"]
y1_bin_14 <- ab_fas_14$Timi23[ab_fas_14$trialid == "2"]
```

```
# patients in the hist and crnt respectively.
n0_14 <- 21
n1_14 <- 227

# prior specifications
btmean.Mu = 0 # for convenience, setting them to be the same
btprec.Mu = 0.1
thtmean.Mu = 0
thtprec.Mu = 0.1

# Pre-specify several borrowing rates.
bttau.alpha <-c(0,0,0);  bttau.beta<-c(0,0,0)
bttau.alpha[1] <- 0.1;  bttau.beta[1] <- 0.1    # lots of borrowing
bttau.alpha[2] <- 40;  bttau.beta[2] <- 4     # a little borrowing
bttau.alpha[3] <- 1000;  bttau.beta[3] <- 10  # not much borrowing


thttau.alpha <-c(0,0,0);  thttau.beta<-c(0,0,0)
thttau.alpha[1] <- 0.1;  thttau.beta[1] <- 0.1    # lots of borrowing
thttau.alpha[2] <- 40;  thttau.beta[2] <- 4     # a little borrowing
thttau.alpha[3] <- 1000;  thttau.beta[3] <- 10  # not much borrowing

# pwrbeta1_14_bin_hier <- rep(0,length(bttau.alpha))
beta1store_14_bin_hier <- NULL
dput(list(mubeta=1, prec=0.1, mutheta=1,
prec0=0.1),"14_bin_hier_inits.txt")
beta1stat_14_bin_hier <- NULL; beta1id <- 0
for (k in 1:length(thttau.alpha)) {

  for (j in 1:length(bttau.alpha)){
    bt1_new <- NULL
    dput(list(x0=x0_bin_14, y0=y0_bin_14, x1=x1_bin_14, y1=y1_bin_14,
n0=n0_14, n1=n1_14,
              btmean.Mu = btmean.Mu, btprec.Mu = btprec.Mu, thtmean.Mu =
thtmean.Mu,
              thtprec.Mu = thtprec.Mu, bttau.alpha = bttau.alpha[j],
bttau.beta = bttau.beta[j],
              thttau.alpha = thttau.alpha[k], thttau.beta =
thttau.beta[k]),"14_bin_hier_data.txt")

    modelCheck("BUGS_binary_hier.txt")
    modelData("14_bin_hier_data.txt")
    modelCompile(numChains=1)
    modelInits("14_bin_hier_inits.txt")
    modelGenInits()

    modelUpdate(5000)
    samplesSet(c("beta1"))
    modelUpdate(50000)

    beta1id <- beta1id + 1
    bt1_new <- cbind(samplesStats("beta1"), bttau.alpha[j], bttau.beta[j],
thttau.alpha[k], thttau.beta[k], beta1id)
    beta1stat_14_bin_hier <- rbind(beta1stat_14_bin_hier, bt1_new)

    # some plots
    # samplesHistory("beta1")
    # samplesDensity("beta1") # Posterior distr.
    beta1store_14_bin_hier <- cbind(beta1store_14_bin_hier,
samplesSample("beta1"))
    # samplesAutoC("beta1", 1)
```

```
    cat("\nstats for beta1:\n"); print(beta1stat_14_bin_hier)

    # cat("\n\nHere are simulated results for beta prior no.",j, "theta
prior no.",k,
    #   ",\n  and true mean_mu, prec_mu, tau_alpha, tau_beta,
theta_tau_alpha, theta_tau_beta are:
",btmean.Mu,btprec.Mu,bttau.alpha[j],bttau.beta[j], thttau.alpha[k],
thttau.beta[k], "\n")
    }
}

# stats for beta1:
#          mean     sd  MC_error val2.5pc median val97.5pc start  sample
bttau.alpha[j] bttau.beta[j] thttau.alpha[k] thttau.beta[k] beta1id
# beta1  0.2056 0.2699 0.0001572  -0.3223 0.2052    0.7364 50001 5000000
2             2               2              2       1
# beta11 0.2049 0.2705 0.0001508  -0.3238 0.2046    0.7368 50001 5000000
6             6               2              2       2
# beta12 0.2045 0.2707 0.0001478  -0.3252 0.2041    0.7367 50001 5000000
16            16               2              2       3

######### BHM Real Hierarchical For 24
########################################

set.seed(9999)

# nrep <- 3

# Bayesian Hierarchical Modeling
# For dataset 24

x0_bin_24 <- ab_fas_24$Group[ab_fas_24$trialid == "1"]
y0_bin_24 <- ab_fas_24$Timi23[ab_fas_24$trialid == "1"]
x1_bin_24 <- ab_fas_24$Group[ab_fas_24$trialid == "2"]
y1_bin_24 <- ab_fas_24$Timi23[ab_fas_24$trialid == "2"]


# patients in the hist and crnt respectively.
n0_24 <- 21
n1_24 <- 228

# prior specifications
btmean.Mu = 0 # for convenience, setting them to be the same
btprec.Mu = 0.1
thtmean.Mu = 0
thtprec.Mu = 0.1


# Pre-specify several borrowing rates.
bttau.alpha <-c(0,0,0);  bttau.beta<-c(0,0,0)
bttau.alpha[1] <- 2;  bttau.beta[1] <- 2    # lots of borrowing
bttau.alpha[2] <- 6;  bttau.beta[2] <- 6     # a little borrowing
bttau.alpha[3] <- 16;  bttau.beta[3] <- 16  # not much borrowing


thttau.alpha <-c(0,0,0);  thttau.beta<-c(0,0,0)
thttau.alpha[1] <- 2;  thttau.beta[1] <- 2    # lots of borrowing
thttau.alpha[2] <- 6;  thttau.beta[2] <- 6     # a little borrowing
thttau.alpha[3] <- 16;  thttau.beta[3] <- 16  # not much borrowing
```

```
# pwrbeta1_24_bin_hier <- rep(0,length(bttau.alpha))
beta1store_24_bin_hier <- NULL
dput(list(mubeta=1, prec=0.1, mutheta=1,
prec0=0.1),"24_bin_hier_inits.txt")
beta1stat_24_bin_hier <- NULL; beta1id <- 0
strt_time <- Sys.time()
for (k in 1:length(thttau.alpha)) {

  for (j in 1:length(bttau.alpha)){
    bt1_new <- NULL
    dput(list(x0=x0_bin_24, y0=y0_bin_24, x1=x1_bin_24, y1=y1_bin_24,
n0=n0_24, n1=n1_24,
              btmean.Mu = btmean.Mu, btprec.Mu = btprec.Mu, thtmean.Mu =
thtmean.Mu,
              thtprec.Mu = thtprec.Mu, bttau.alpha = bttau.alpha[j],
bttau.beta = bttau.beta[j],
              thttau.alpha = thttau.alpha[k], thttau.beta =
thttau.beta[k]),"24_bin_hier_data.txt")

    modelCheck("BUGS_binary_hier.txt")
    modelData("24_bin_hier_data.txt")
    modelCompile(numChains=4)
    modelInits("24_bin_hier_inits.txt")
    modelGenInits()

    modelUpdate(2000)
    samplesSet(c("beta1"))
    modelUpdate(5000)

    beta1id <- beta1id + 1
    bt1_new <- cbind(samplesStats("beta1"), bttau.alpha[j], bttau.beta[j],
thttau.alpha[k], thttau.beta[k], beta1id)
    beta1stat_24_bin_hier <- rbind(beta1stat_24_bin_hier, bt1_new)

    samplesDensity("beta1") # Posterior distr.
    beta1store_24_bin_hier <- cbind(beta1store_24_bin_hier,
samplesSample("beta1"))

    cat("\nstats for beta1:\n"); print(beta1stat_24_bin_hier)
    }
}
TTtime_est <- strt_time - Sys.time()
# stats for beta1:
#          mean     sd  MC_error val2.5pc median val97.5pc start  sample
bttau.alpha[j] bttau.beta[j] thttau.alpha[k] thttau.beta[k] beta1id
# beta1  0.5531 0.2785 0.0001647 0.010120 0.5519     1.103 50001 5000000
2              2               2              2       1
# beta11 0.5516 0.2791 0.0001609 0.007483 0.5506     1.102 50001 5000000
6              6               2              2       2
# beta12 0.5515 0.2794 0.0001579 0.007417 0.5502     1.103 50001 5000000
16            16               2              2       3


######### BHM for the original dataset(Hist) #################

# For 14 Hist

dput(list(mu=1, prec=0.1, mu0=1, prec0=0.1),"14_hist_inits.txt")
beta1stat_14_hist <- NULL
k=2; j=2
bt1_new <- NULL
```

```
dput(list(x=x0_bin_14,y=y0_bin_14,n=n0_14,mean.Mu=mean_mu,prec.Mu=prec_mu,
          mean.Mu0=mean_mu0,prec.Mu0=prec_mu0,
          tau.alpha=as.numeric(tau_alpha[j]),
          tau.beta=as.numeric(tau_beta[j]),
          tau.alpha0=as.numeric(tau_alpha0[k]),
          tau.beta0=as.numeric(tau_beta0[k])),"14_hist_data.txt")

modelCheck("BUGS_binary.txt")
modelData("14_hist_data.txt")
modelCompile(numChains=1)
modelInits("14_hist_inits.txt")
modelGenInits()

modelUpdate(50000)
samplesSet(c("beta1"))
modelUpdate(5000000) # 50000 is not enough for convergence.
bt1_new <- cbind(samplesStats("beta1"), tau_alpha[j], tau_beta[j],
tau_alpha0[k], tau_beta0[k])
beta1stat_14_hist <- rbind(beta1stat_14_hist, bt1_new)
beta1stat_14_hist
#         mean     sd  MC_error val2.5pc median val97.5pc start  sample
# beta1 0.1753 0.9689 0.0009168   -1.722 0.1712     2.094 50001 5000000
#       tau_alpha[j] tau_beta[j] tau_alpha0[k] tau_beta0[k]
# beta1            6           6            16           16


# some plots
# samplesHistory("beta1")
# samplesDensity("beta1") # Posterior distr.
# samplesAutoC("beta1", 1)

# For 24 Hist

dput(list(mu=1, prec=0.1, mu0=1, prec0=0.1),"24_hist_inits.txt")
beta1stat_24_hist <- NULL
k=2; j=2
bt1_new <- NULL
dput(list(x=x0_bin_24,y=y0_bin_24,n=n0_24,mean.Mu=mean_mu,prec.Mu=prec_mu,
          mean.Mu0=mean_mu0,prec.Mu0=prec_mu0,
          tau.alpha=as.numeric(tau_alpha[j]),
          tau.beta=as.numeric(tau_beta[j]),
          tau.alpha0=as.numeric(tau_alpha0[k]),
          tau.beta0=as.numeric(tau_beta0[k])),"24_hist_data.txt")

modelCheck("BUGS_binary.txt")
modelData("24_hist_data.txt")
modelCompile(numChains=1)
modelInits("24_hist_inits.txt")
modelGenInits()

modelUpdate(50000)
samplesSet(c("beta1"))
modelUpdate(5000000) # 50000 is not enough for convergence.
bt1_new <- cbind(samplesStats("beta1"), tau_alpha[j], tau_beta[j],
tau_alpha0[k], tau_beta0[k])
beta1stat_24_hist <- rbind(beta1stat_24_hist, bt1_new)
beta1stat_24_hist
#         mean    sd MC_error val2.5pc median val97.5pc start  sample
# beta1 0.7235 1.054 0.001081   -1.287 0.7002      2.87 50001 5000000
#       tau_alpha[j] tau_beta[j] tau_alpha0[k] tau_beta0[k]
# beta1            6           6            16           16
```

```
# For 14 Curnt
dput(list(mu=1, prec=0.1, mu0=1, prec0=0.1),"14_crnt_inits.txt")
beta1stat_14_crnt <- NULL
k=2; j=2
bt1_new <- NULL
dput(list(x=x1_bin_14,y=y1_bin_14,n=n1_14,mean.Mu=mean_mu,prec.Mu=prec_mu,
          mean.Mu0=mean_mu0,prec.Mu0=prec_mu0,
          tau.alpha=as.numeric(tau_alpha[j]),
          tau.beta=as.numeric(tau_beta[j]),
          tau.alpha0=as.numeric(tau_alpha0[k]),
          tau.beta0=as.numeric(tau_beta0[k])),"14_crnt_data.txt")

modelCheck("BUGS_binary.txt")
modelData("14_crnt_data.txt")
modelCompile(numChains=1)
modelInits("14_crnt_inits.txt")
modelGenInits()

modelUpdate(50000)
samplesSet(c("beta1"))
modelUpdate(5000000) # 50000 is not enough for convergence.
bt1_new <- cbind(samplesStats("beta1"), tau_alpha[j], tau_beta[j],
tau_alpha0[k], tau_beta0[k])
beta1stat_14_crnt <- rbind(beta1stat_14_crnt, bt1_new)
beta1stat_14_crnt
#         mean     sd  MC_error val2.5pc median val97.5pc start  sample
tau_alpha[j] tau_beta[j] tau_alpha0[k] tau_beta0[k]
# beta1 0.2126 0.2743 0.0001534  -0.3242  0.212    0.7514 50001 5000000
6           6             6            6


# For 24 Crnt
dput(list(mu=1, prec=0.1, mu0=1, prec0=0.1),"24_crnt_inits.txt")
beta1stat_24_crnt <- NULL
k=2; j=2
bt1_new <- NULL
dput(list(x=x1_bin_24,y=y1_bin_24,n=n1_24,mean.Mu=mean_mu,prec.Mu=prec_mu,
          mean.Mu0=mean_mu0,prec.Mu0=prec_mu0,
          tau.alpha=as.numeric(tau_alpha[j]),
          tau.beta=as.numeric(tau_beta[j]),
          tau.alpha0=as.numeric(tau_alpha0[k]),
          tau.beta0=as.numeric(tau_beta0[k])),"24_crnt_data.txt")

modelCheck("BUGS_binary.txt")
modelData("24_crnt_data.txt")
modelCompile(numChains=1)
modelInits("24_crnt_inits.txt")
modelGenInits()

modelUpdate(50000)
samplesSet(c("beta1"))
modelUpdate(5000000) # 50000 is not enough for convergence.
bt1_new <- cbind(samplesStats("beta1"), tau_alpha[j], tau_beta[j],
tau_alpha0[k], tau_beta0[k])
beta1stat_24_crnt <- rbind(beta1stat_24_crnt, bt1_new)
beta1stat_24_crnt
#         mean    sd  MC_error val2.5pc median val97.5pc start  sample
tau_alpha[j] tau_beta[j] tau_alpha0[k] tau_beta0[k]
# beta1 0.5534 0.283 0.0001572 0.001305 0.5522     1.111 50001 5000000
6           6             6            6
```

```
######### Bayesian (& Normalized) Power Prior for 14 + 24
#####################

# Bayesian Power Prior

# Reference: Carvalho, LM, Ibrahim, JG. On the normalized power prior.
Statistics in Medicine. 2021; 40: 5251–5275. https://doi.org/10.1002/sim.
9124
######### sub Global value ##############

set.seed(9999)
rstan_options(auto_write = TRUE)
options(mc.cores = 4)
P <- 1 # Only the "Group" variable

# Initialization
as <- .5
bs <- 2
vb <- 1.5

eta <- 1
nu <- 1


######### sub For 14 #############

N_Hist_PP_14 <- 21
X_Hist_PP_14 <- matrix(NA, nrow = N_Hist_PP_14, ncol = P)
X_Hist_PP_14 <- ab_fas_14$Group[ab_fas_14$trialid == "1"]
Y_Hist_PP_14 <- ab_fas_14$Timi23[ab_fas_14$trialid == "1"]
X_Hist_PP_14 <- as.matrix(X_Hist_PP_14)


### will draw from the same process for simplicity
N_Cur_PP_14 <- 227
X_Cur_PP_14 <- matrix(NA, nrow = N_Cur_PP_14, ncol = P)
X_Cur_PP_14 <- ab_fas_14$Group[ab_fas_14$trialid == "2"]
Y_Cur_PP_14 <- ab_fas_14$Timi23[ab_fas_14$trialid == "2"]
X_Cur_PP_14 <- as.matrix(X_Cur_PP_14)

###
#scenario <- "B"
#source(paste("data_logistic_regression_scenario_", scenario, ".r", sep =
"""))

summary(glm(Y_Cur_PP_14 ~ X_Cur_PP_14, family = "binomial"))
# summary(glm(y ~ X, family = "binomial")) # unknown the true value
# summary(glm(c(y_0, y) ~ rbind(X_0, X), family = "binomial"))
# Treatment effect: 0.2107(SE: 0.2741), p-value = 0.4421


lgr.data.forposterior <- list(
  N0 = N_Hist_PP_14,
  X0 = X_Hist_PP_14,
  P = P,
  y0 = Y_Hist_PP_14,
  eta = eta,
  nu = nu,
  X = X_Cur_PP_14,
  N = N_Cur_PP_14,
```

```
  y = Y_Cur_PP_14,
  a_0 = 0.99
)

lgr.fora0.data <- list(
  N0 = N_Hist_PP_14,
  X0 = X_Hist_PP_14,
  P = P,
  y0 = Y_Hist_PP_14,
  a_0 = NULL
)

# Unnormalised PP(iter: 50000; warm_up: 5000)
#                  mean      se_mean         sd          2.5%           25%
50%          75%        97.5%     n_eff     Rhat
# alpha      0.39707349 0.0006298052 0.19122155  2.392479e-02    0.26770105
0.39587747    0.5249478    0.7761060  92185.18 1.000026
# beta[1]    0.21298564 0.0009074029 0.27358197 -3.225526e-01    0.02815285
0.21269092    0.3980267    0.7504188  90902.25 1.000019
# a_0        0.07681133 0.0002026846 0.07666139  1.887284e-03    0.02211084
0.05337327    0.1065202    0.2819819 143057.94 1.000008
# lp__     -166.59340291 0.0050352809 1.29105159 -1.699292e+02 -167.18487159
-166.25950751 -165.6485663 -165.1189724  65741.53 1.000019

# app normalised 14 (Iter:50000, warm_up:5000)

# [[1]] Ks=20
#                  mean      se_mean        sd          2.5%           25%
50%          75%        97.5%    n_eff     Rhat
# alpha      0.4159704 0.0006534947 0.1867658    0.05401632    0.28897783
0.4152217    0.5416334    0.7864004 81679.03 1.000066
# beta[1]    0.2125040 0.0009353397 0.2676888   -0.30974278    0.03161997
0.2124524    0.3917383    0.7408682 81907.15 1.000101
# a_0        0.6416362 0.0007211028 0.2263360    0.16685873    0.47710657
0.6748259    0.8336658    0.9600451 98517.38 1.000074
# lp__     -155.1125815 0.0043219099 1.1663429 -158.13855756 -155.64179827
-154.8302903 -154.2629935 -153.7565368 72828.46 1.000082
#
# [[2]] Ks=100
#                  mean      se_mean        sd          2.5%           25%
50%          75%        97.5%     n_eff      Rhat
# alpha      0.4183601 0.0006014189 0.1866289    0.05460952    0.29218151
0.4174768    0.5431971    0.7879330  96294.99 0.9999972
# beta[1]    0.2089681 0.0008645815 0.2671475   -0.31410989    0.02984737
0.2085931    0.3887734    0.7333530  95475.18 0.9999894
# a_0        0.6540267 0.0006679198 0.2325524    0.16834770    0.48348185
0.6879347    0.8518415    0.9810772 121225.21 0.9999960
# lp__     -155.1942884 0.0044915779 1.2282051 -158.30531844 -155.78224362
-154.8884668 -154.2821401 -153.7600255  74772.84 1.0000516
#
# [[3]] Ks=1000
#                  mean      se_mean        sd          2.5%           25%
50%          75%        97.5%     n_eff      Rhat
# alpha      0.4173179 0.0006118890 0.1870573    0.05547516    0.29037879
0.4163234    0.5424399    0.7875703  93455.30 1.0000057
# beta[1]    0.2109450 0.0008718313 0.2683710   -0.31446166    0.02917057
0.2106142    0.3910608    0.7373050  94755.88 1.0000215
# a_0        0.6587530 0.0006673675 0.2335305    0.16800019    0.48914109
0.6939756    0.8568057    0.9864564 122449.55 0.9999974
# lp__     -155.2303142 0.0048233420 1.2755796 -158.50125828 -155.82201404
-154.9036492 -154.2911738 -153.7588287  69938.94 1.0000339#
```

```
#
# [[4]] Ks=10000
#                 mean      se_mean        sd          2.5%           25%
50%          75%        97.5%     n_eff     Rhat
# alpha      0.4178922 0.0005979232 0.1862345    0.05696694    0.29085259
0.4162044    0.5429623    0.7867345  97012.88 1.000007
# beta[1]    0.2103730 0.0008608620 0.2671250   -0.30982465    0.02978204
0.2100716    0.3900134    0.7341805  96285.78 1.000025
# a_0        0.6577112 0.0006723257 0.2335368    0.16765356    0.48772210
0.6916880    0.8561546    0.9861495 120656.60 1.000027
# lp__     -155.2224532 0.0048064735 1.2749075 -158.49600552 -155.80650066
-154.8957674 -154.2876287 -153.7599599  70356.50 1.000014

######### sub For 24 ##################

N_Hist_PP_24 <- 21
X_Hist_PP_24 <- matrix(NA, nrow = N_Hist_PP_24, ncol = P)
X_Hist_PP_24 <- ab_fas_24$Group[ab_fas_24$trialid == "1"]
Y_Hist_PP_24 <- ab_fas_24$Timi23[ab_fas_24$trialid == "1"]
X_Hist_PP_24 <- as.matrix(X_Hist_PP_24)

### will draw from the same process for simplicity
N_Cur_PP_24 <- 228
X_Cur_PP_24 <- matrix(NA, nrow = N_Cur_PP_24, ncol = P)
X_Cur_PP_24 <- ab_fas_24$Group[ab_fas_24$trialid == "2"]
Y_Cur_PP_24 <- ab_fas_24$Timi23[ab_fas_24$trialid == "2"]
X_Cur_PP_24 <- as.matrix(X_Cur_PP_24)

sum(Y_Hist_PP_14)

lgr.data.forposterior <- list(
  N0 = N_Hist_PP_24,
  X0 = X_Hist_PP_24,
  P = P,
  y0 = Y_Hist_PP_24,
  eta = eta,
  nu = nu,
  X = X_Cur_PP_24,
  N = N_Cur_PP_24,
  y = Y_Cur_PP_24,# a_0 can be removed together with the comma
  a_0 = 0.5
)

lgr.fora0.data <- list(
  N0 = N_Hist_PP_24,
  X0 = X_Hist_PP_24,
  P = P,
  y0 = Y_Hist_PP_24,
  a_0 = NULL
)

#
# summary(glm(Y_Hist_PP_24 ~ X_Hist_PP_24, family = "binomial"))
# confint(glm(Y_Hist_PP_24 ~ X_Hist_PP_24, family = "binomial"))
# Treatment effect: 0.6568(SE: 1.0427), p-value = 0.529
#                   2.5 %   97.5 %
# (Intercept)  -0.4328021 2.382047
# X_Hist_PP_24 -1.3807395 2.886597


# Unnormalized PP (iter:50000,warm_up=5000)
```

```
#                 mean      se_mean         sd          2.5%           25%
50%          75%        97.5%     n_eff      Rhat
# alpha      0.3981754 0.0006199904 0.19051378  2.932755e-02    0.26877254
0.39701810    0.5258808    0.7753177  94424.09 1.0000037
# beta[1]    0.5559549 0.0009206240 0.28235186  5.949202e-03    0.36508944
0.55437951    0.7463626    1.1119831  94062.55 0.9999953
# a_0        0.0841467 0.0002250911 0.08416756  2.117972e-03    0.02439461
0.05829959    0.1166441    0.3127097 139821.19 1.0000112
# lp__     -160.7456446 0.0049572595 1.28828444 -1.640827e+02 -161.33537055
-160.41147948 -159.8056652 -159.2737763  67536.76 1.0000473

# Approximate NPP(50000;5000)
# [[1]] Ks=20
#                 mean      se_mean         sd          2.5%           25%
50%          75%        97.5%     n_eff      Rhat
# alpha      0.4166120 0.0006105680 0.1874779    0.05206738    0.2898097
0.4156787    0.5421286    0.7869005  94282.75 1.000004
# beta[1]    0.5619864 0.0009092634 0.2774500    0.02227581    0.3744867
0.5609924    0.7479481    1.1092731  93108.65 1.000017
# a_0        0.6383430 0.0006751730 0.2255710    0.16699300    0.4731491
0.6699296    0.8297373    0.9596292 111618.54 1.000020
# lp__     -149.5901131 0.0042610903 1.1715688 -152.61919506 -150.1234056
-149.3034978 -148.7346721 -148.2324045  75595.18 1.000030
# [[2]] Ks=100
#                 mean      se_mean         sd          2.5%           25%
50%          75%        97.5%     n_eff      Rhat
# alpha      0.4176921 0.0005931203 0.1876985    0.05450538    0.2904976
0.4169927    0.5428828    0.7897754 100146.59 0.9999959
# beta[1]    0.5610284 0.0008750438 0.2773148    0.02241920    0.3728049
0.5609640    0.7468796    1.1065646 100435.33 0.9999948
# a_0        0.6505448 0.0006597572 0.2321004    0.16812537    0.4797539
0.6827171    0.8472835    0.9810079 123760.91 0.9999978
# lp__     -149.6688930 0.0044760832 1.2273257 -152.78225016 -150.2601960
-149.3621009 -148.7587176 -148.2337409  75183.65 1.0000447
#
# [[3]] Ks=1000
#                 mean      se_mean         sd          2.5%           25%
50%          75%        97.5%     n_eff      Rhat
# alpha      0.4178711 0.0006100458 0.1870108    0.05547070    0.2915026
0.4172983    0.5425744    0.7871074  93974.23 1.0000024
# beta[1]    0.5618961 0.0008965964 0.2766949    0.02255433    0.3748876
0.5606280    0.7477762    1.1052205  95237.56 0.9999917
# a_0        0.6544113 0.0006678186 0.2329232    0.16844812    0.4840482
0.6866557    0.8523486    0.9855371 121648.98 0.9999965
# lp__     -149.6946936 0.0046893868 1.2715654 -152.95708744 -150.2825787
-149.3671713 -148.7604652 -148.2343890  73526.73 1.0000091
# [[4]] Ks=10000
#                 mean      se_mean         sd          2.5%           25%
50%          75%        97.5%     n_eff      Rhat
# alpha      0.4174263 0.0005984056 0.1877485    0.05319848    0.2901402
0.4158094    0.5431475    0.7890770  98437.70 0.9999975
# beta[1]    0.5626029 0.0008829917 0.2773858    0.02265995    0.3753349
0.5609180    0.7493258    1.1081355  98685.89 1.0000134
# a_0        0.6542221 0.0006721948 0.2330486    0.16995951    0.4834836
0.6861853    0.8519940    0.9857745 120199.48 1.0000053
# lp__     -149.7073997 0.0050004420 1.2904660 -153.04742877 -150.2964540
-149.3722395 -148.7643575 -148.2338757  66600.33 1.0000120

######### sub Global function ######

### 1. Unnormalised posterior
```

```
Unnormalized_PP_posterior <- function(model, data, iter=50000, warmup=5000)
{

    compiled.model.unnorm.posterior <- stan_model(model)

    unnorm.posterior.lgr <- sampling(compiled.model.unnorm.posterior, data
= data, iter = iter, warmup = warmup)
    rslt <- summary(unnorm.posterior.lgr)
    return(rslt)
}
rslt <- Unnormalized_PP_posterior(model =
"simple_logistic_regression_posterior_unnormalised.stan",
                                  data = lgr.data.forposterior, iter =
50000)
rslt

### 2. Specify \alpha manually
Manually_a_Unnormalized_PP_posterior <- function(model, data, iter=50000,
warmup=5000){

  compiled.model.unnorm.posterior <- stan_model(model)

  unnorm.posterior.lgr <- sampling(compiled.model.unnorm.posterior, data =
data, iter = iter, warmup = warmup)
  rslt <- summary(unnorm.posterior.lgr)
  return(rslt)
}
rslt <- Manually_a_Unnormalized_PP_posterior(model = "NPP_Manually_a.stan",
                                  data = lgr.data.forposterior, iter =
50000)
rslt



### 3. The approximately normalised prior
get_posterior_K <- function(K, data_postrr, constt_data, model,
approx_model, iter = 13000, warmup = 3000){
  pred_a0s <- seq(0, max(constt_data$a0), length.out = K) # splitting
points
  a0_grid <- data.frame(a0 = pred_a0s,
                        lc_pred = predict(approx_model, newdata =
data.frame(a0 = pred_a0s)))
  data_postrr$pred_grid_x <- a0_grid$a0
  data_postrr$pred_grid_y <- a0_grid$lc_pred
  approx.norm.posterior.lgr <- sampling(model,
                                        data = data_postrr, refresh = 0,
                                        iter = iter, warmup = warmup)
  sumry_stat <- summary(approx.norm.posterior.lgr)
  cat('posterior generated\n')
  return(sumry_stat)
}

Approximation_Normalized_PP <- function(J=20, maxA=1, epsilon=0.05,
prior_stancode,
                                        postrr_stancode, data_a0,
data_postrr, Ks=5000){

  logistic_prior <- stan_model(prior_stancode)
  adaptive.ca0.estimates <- build_grid(compiled.model.prior =
logistic_prior, eps = epsilon,
```

```
                                        M = maxA, J = J, v1 = 10, v2 = 10,
                                        stan.list = data_a0, pars =
c("alpha", "beta"))
  constant_data <- adaptive.ca0.estimates$result

  fit.gam <- mgcv::gam(lc_a0 ~ s(a0, k = J + 1), data = constant_data) #
Spline smoothing function Generalized Additive Model.
  cat('approximate constant data generated\n')
  approx.normalised.model <- stan_model(postrr_stancode)

  all.approximates <- get_posterior_K(K=Ks, model=approx.normalised.model,
                                      constt_data=constant_data,
data_postrr = data_postrr, approx_model = fit.gam)
  return(all.approximates)
}

Approximation_Normalized_PP(prior_stancode =
"logistic_regression_prior.stan",
                            postrr_stancode = "NPP_approximate.stan",
                            data_a0 = lgr.fora0.data,
                            data_postrr = lgr.data.forposterior)

################# plot: Analysis of p(a_0 | data)#################

compiled.model.unnorm.posterior <- stan_model("Unnormal_PP.stan")

unnorm.posterior.lgr <- sampling(compiled.model.unnorm.posterior, data =
lgr.data.forposterior, iter = 50000, warmup = 5000)

# unnorm.posterior.lgr
pairs(unnorm.posterior.lgr, pars = c("a_0", "beta"))
logistic_prior <- stan_model("logistic_regression_prior.stan")
adaptive.ca0.estimates <- build_grid(compiled.model.prior = logistic_prior,
eps = 0.05,
                                     M = 1, J = 20, v1 = 10, v2 = 10,
                                     stan.list = lgr.fora0.data, pars =
c("alpha", "beta"))
constant_data <- adaptive.ca0.estimates$result

fit.gam <- mgcv::gam(lc_a0 ~ s(a0, k = J + 1), data = constant_data) #
Spline smoothing function Generalized Additive Model.
approx.normalised.model <- stan_model("NPP_approximate.stan")

Ks <- c(20, 1000, 5000, 10000)

all.approximates <- lapply(Ks, function(K) {
  get_posterior_K(constt_data = constant_data,
                  data_postrr = lgr.data.forposterior,
                  model = approx.normalised.model,
                  approx_model = fit.gam,
                  K = K)
})

a0.unnorm <- extract(unnorm.posterior.lgr, 'a_0')$a_0

a0.approx.list <- lapply(Ks, function(k){
  i <- match(k, Ks)
  a0 <- extract(all.approximates[[i]], 'a_0')$a_0
  return(data.frame(a0 = a0, normalisation = paste("K=", k, sep = "")))
} )
```

```
a0.dt <-
  rbind(
    data.frame(a0 = a0.unnorm,
               normalisation = rep("none", length(a0.unnorm))
    ),
    do.call(rbind, a0.approx.list)
  )

######### PP Generating plots ###########################################

library(ggplot2)

a0.dt$normalisation <- factor(a0.dt$normalisation,
                              levels = c("none", paste("K=", Ks, sep = ""))
)


a0_dist <- ggplot(a0.dt, aes(x = a0, fill = normalisation, colour =
normalisation)) +
  geom_density() +
  stat_function(fun = function(x) dbeta(x, eta, nu),
                geom = "line", colour = "black", linetype = "longdash") +
  ggtitle("Logistic regression") +
  facet_grid(normalisation~., scales = "free") +
  scale_y_continuous("Density", expand = c(0, 0)) +
  scale_x_continuous(expression(a[0]), expand = c(0, 0)) +
  theme_bw(base_size = 20) +
  theme(legend.position = "bottom",
        legend.justification = "centre",
        legend.title = element_blank(),
        strip.background = element_blank(),
        strip.text.y = element_blank(),
        legend.margin = margin(0, 0, 0, 0),
        legend.box.margin = margin(0, 0, 0, 0))

a0_dist
# For 14
# ggsave(paste("../../figures/a0_posterior_RegressionLogistic_J=", J,
"_scenario_", scenario, ".pdf", sep = ""), a0_dist)

# For 24
# ggsave(paste("a0_24_posterior_RegressionLogistic_J=", J, "_scenario_",
scenario, ".pdf", sep = ""), a0_dist)



a0_dist_norm <- ggplot(subset(a0.dt, normalisation != "none"),
                       aes(x = normalisation, y = a0,
                           fill = normalisation, colour = normalisation)) +
  geom_boxplot(alpha = .4) +
  ggtitle("Logistic regression") +
  scale_y_continuous(expression(a[0]), expand = c(0, 0)) +
  theme_bw(base_size = 20)

a0_dist_norm

# For 14
#
ggsave(paste("a0_14_posterior_RegressionLogistic_normalisation_comparison_J=",
J, "_scenario_", scenario, ".pdf", sep = ""), a0_dist_norm)
```

```
# For 24
#
ggsave(paste("a0_24_posterior_RegressionLogistic_normalisation_comparison_J=",
J, "_scenario_", scenario, ".pdf", sep = ""), a0_dist_norm)

###
unnorm.pars <- as.data.frame(
  cbind(extract(unnorm.posterior.lgr, 'alpha')$alpha,
        extract(unnorm.posterior.lgr, 'beta')$beta)
)
names(unnorm.pars) <- c("alpha", paste("beta[", 1, "]", sep = "")) # Only
one covariate(Group) here.

##

app.norm.pars.list <- lapply(all.approximates, function(x) {
  pars <- as.data.frame(
    cbind(extract(x, 'alpha')$alpha,
          extract(x, 'beta')$beta
    )
  )
  names(pars) <- c("alpha", paste("beta[", 1, "]", sep = "")) # Only one
covariate here.
  return(pars)
})

library(reshape2)
app.norm.pars.df.list <- lapply(seq_along(app.norm.pars.list), function(i)
{
  y <- app.norm.pars.list[[i]]
  dt <- melt(y, variable.name = "parameter", value.name = "sample")
  dt$normalisation <- paste("K=", Ks[i] ,sep = "")
  return(dt)
} )

##

unnorm.dt <- melt(unnorm.pars, variable.name = "parameter", value.name =
"sample")
unnorm.dt$normalisation <- "none"

posterior.dt <- rbind(unnorm.dt, do.call(rbind, app.norm.pars.df.list))
posterior.dt$normalisation <- factor(posterior.dt$normalisation,
                                     levels = c("none", paste("K=", Ks, sep
= "")) )

###

# true.pars <- data.frame(parameter = names(unnorm.pars), value =
c(true.alpha, true.betas))

parameter_posteriors <- ggplot(data = posterior.dt, aes(x = sample, colour
= normalisation, fill = normalisation)) +
  geom_density(alpha = .4) +
  scale_x_continuous("") +
  scale_y_continuous("Density", expand = c(0, 0)) +
  facet_wrap(.~parameter,  scales = "free", labeller = label_parsed) +
  theme_bw(base_size = 20) +
  theme(legend.position = "bottom",
        legend.justification = "centre",
        legend.title = element_blank(),
```

```
        strip.background = element_blank(),
        strip.text.y = element_blank(),
        legend.margin = margin(0, 0, 0, 0),
        legend.box.margin = margin(0, 0, 0, 0))

parameter_posteriors

# For 14
# ggsave(paste("14+parameter_posterior_LogisticRegression_J=", J,
#              "_scenario_", scenario, ".pdf", sep = ""), plot =
parameter_posteriors, dpi = 400)

# For 24
# ggsave(paste("24+parameter_posterior_LogisticRegression_J=", J,
#              "_scenario_", scenario, ".pdf", sep = ""), plot =
parameter_posteriors, dpi = 400)


######### Commensurate Power Prior ################

# Data preparation for 14 (exactly the same)
P <- 1
N_Hist_CPP_14 <-21
X_Hist_CPP_14 <- matrix(NA, nrow = N_Hist_CPP_14, ncol = P)
X_Hist_CPP_14 <- ab_fas_14$Group[ab_fas_14$trialid == "1"]
Y_Hist_CPP_14 <- ab_fas_14$Timi23[ab_fas_14$trialid == "1"]
X_Hist_CPP_14 <- as.matrix(X_Hist_CPP_14)

N_Cur_CPP_14 <- 227
X_Cur_CPP_14 <- matrix(NA, nrow = N_Cur_CPP_14, ncol = P)
X_Cur_CPP_14 <- ab_fas_14$Group[ab_fas_14$trialid == "2"]
Y_Cur_CPP_14 <- ab_fas_14$Timi23[ab_fas_14$trialid == "2"]
X_Cur_CPP_14 <- as.matrix(X_Cur_CPP_14)

CPP_lgr_data <- list (
  N0 = N_Hist_CPP_14,
  X0 = X_Hist_CPP_14,
  P = P,
  y0 = Y_Hist_CPP_14,
  X1 = X_Cur_CPP_14,
  N1 = N_Cur_CPP_14,
  y1 = Y_Cur_CPP_14
)
# W/out cov

#                   mean      se_mean          sd          2.5%
25%           50%          75%        97.5%    n_eff      Rhat
# alpha0      0.50136006 0.0020774466 0.81478418 -1.115286e+00
-0.03059670    0.50216161    1.0371367    2.1037584 153824.5 1.0000219
# alpha1      0.39732028 0.0005870756 0.19146623  2.460464e-02
0.26768938    0.39663951    0.5260455    0.7743442 106364.4 1.0000041
# beta0[1]    0.25905225 0.0023107348 0.87547172 -1.466288e+00
-0.32022974    0.25763118    0.8347469    2.0004766 143543.7 0.9999939
# beta1[1]    0.21269958 0.0008627126 0.27569203 -3.259791e-01
0.02651864    0.21171937    0.3985240    0.7542235 102121.3 0.9999988
# a_0         0.09634682 0.0002468944 0.09090458  3.634976e-03
0.03158143    0.06992141    0.1326496    0.3364320 135565.4 1.0000114
# tau         0.87364957 0.0002913760 0.10404757  6.152454e-01
0.81639244    0.89952227    0.9557303    0.9958813 127513.5 1.0000255
# lp__      -172.42933930 0.0073334824 1.88503020 -1.770098e+02
-173.43707897 -172.08091062 -171.0435441 -169.8162721  66071.8 1.0000299
```

```
# Data preparation for 24
P <- 1
N_Hist_CPP_24 <-21
X_Hist_CPP_24 <- matrix(NA, nrow = N_Hist_CPP_24, ncol = P)
X_Hist_CPP_24 <- ab_fas_24$Group[ab_fas_24$trialid == "1"]
Y_Hist_CPP_24 <- ab_fas_24$Timi23[ab_fas_24$trialid == "1"]
X_Hist_CPP_24 <- as.matrix(X_Hist_CPP_24)

N_Cur_CPP_24 <- 228
X_Cur_CPP_24 <- matrix(NA, nrow = N_Cur_CPP_24, ncol = P)
X_Cur_CPP_24 <- ab_fas_24$Group[ab_fas_24$trialid == "2"]
Y_Cur_CPP_24 <- ab_fas_24$Timi23[ab_fas_24$trialid == "2"]
X_Cur_CPP_24 <- as.matrix(X_Cur_CPP_24)

CPP_lgr_data <- list (
  N0 = N_Hist_CPP_24,
  X0 = X_Hist_CPP_24,
  P = P,
  y0 = Y_Hist_CPP_24,
  X1 = X_Cur_CPP_24,
  N1 = N_Cur_CPP_24,
  y1 = Y_Cur_CPP_24
)

# W/out cov

#                  mean      se_mean        sd          2.5%           25%
50%          75%        97.5%     n_eff      Rhat
# alpha0      0.5207578 0.0021639522 0.8092597 -1.087205e+00 -6.800769e-03
0.52150921    1.0480837    2.1243475 139855.80 1.0000280
# alpha1      0.3973456 0.0005851908 0.1904246  2.568358e-02  2.687694e-01
0.39645329    0.5242410    0.7740165 105889.09 1.0000418
# beta0[1]    0.6255416 0.0023442393 0.8710771 -1.089022e+00  4.999784e-02
0.62634787    1.2011088    2.3473034 138073.18 0.9999952
# beta1[1]    0.5571051 0.0008734638 0.2823780  6.775774e-03  3.661976e-01
0.55540583    0.7465752    1.1146200 104513.36 1.0000248
# a_0         0.1093766 0.0002918065 0.1038825  4.221536e-03  3.575692e-02
0.07898803    0.1505966    0.3848436 126734.48 1.0000155
# tau         0.8685565 0.0003060471 0.1082847  6.016592e-01  8.088691e-01
0.89555829    0.9541231    0.9958456 125186.52 1.0000217
# lp__      -166.5242074 0.0072120238 1.8726576 -1.710549e+02 -1.675327e+02
-166.18214360 -165.1473552 -163.9156309  67422.12 1.0000538



# Commensurate PP posterior(w/out cov)
CPP_get_posterior <- function(stancode, data, iter=50000, warmup=5000){
  CPP_posterior <- stan_model(stancode)
  CPP_posterior_lgr <- sampling(CPP_posterior, data=data, iter=iter,
warmup=warmup, chains=4)
  rslt <- summary(CPP_posterior_lgr)
  return(rslt)
}
CPP_get_posterior(stancode = "CPP_wout_cov.stan", data = CPP_lgr_data, iter
= 50000, warmup = 5000 )


######### Preparation: Virtual Dataset ################
```

```
######### Fake-data simulation ######################

# Reference: Gelman, A. and J. Hill (2006). "Data Analysis Using Regression
And Multilevel/Hierarchical Models." Cambridge Universty Press 3.
# Fake data generating function.

fake <- function(Hist, Crnt, Histprob0, Histprob1, Crntprob0, Crntprob1){
  trialid <- c(rep(1, Hist), rep(2, Crnt)) # the same as the 2-4 of a+b

  Hist_Cntrl <- round(Hist/2)
  Hist_Trt <- ceiling(Hist/2)
  Crnt_Cntrl <- round(Crnt/2)
  Crnt_Trt <- ceiling(Crnt/2)

  Tm23_Hist_Cntrl <- c(rbinom(Hist_Cntrl, 1, Histprob0))
  Tm23_Hist_Trt <- c(rbinom(Hist_Trt, 1, Histprob1))
  Tm23_Crnt_Cntrl <- c(rbinom(Crnt_Cntrl, 1, Crntprob0))
  Tm23_Crnt_Trt <- c(rbinom(Crnt_Trt, 1, Crntprob1))
  # Grp <- c(Group_Hist, Group_Crnt)
  Group <- c(rep(0, Hist_Cntrl), rep(1, Hist_Trt), rep(0, Crnt_Cntrl),
rep(1, Crnt_Trt))
  Timi23 <- c(Tm23_Hist_Cntrl, Tm23_Hist_Trt, Tm23_Crnt_Cntrl,
Tm23_Crnt_Trt)

  return(data.frame(Timi23, Group, trialid))
}

set.seed(9999)
# Scenario 1: similar to the combination of 24

fake_s1 <- fake(Hist = 21, Crnt = 228, Histprob0 = 0.7, Crntprob0 = 0.6,
Histprob1 = 0.82, Crntprob1 = 0.72)
fake_s2 <- fake(Hist = 21, Crnt = 228, Histprob0 = 0.7, Crntprob0 = 0.6,
Histprob1 = 0.82, Crntprob1 = 0.6)

######### Fake-data GLM Power ######################################
set.seed(9999)
# 0. Chi-square test
power.chisq.test(n=248, w=0.1, df=1, sig.level = 0.05, power = NULL) #
power = 0.3502626
power.chisq.test(n=248, w=0.12, df=1, sig.level = 0.05, power = NULL) #
power = 0.4720755
power.chisq.test(n=248, w=0.15, df=1, sig.level = 0.05, power = NULL) #
power = 0.6562534
power.chisq.test(n=248, w=0.3, df=1, sig.level = 0.05, power = NULL) #
power = 0.997149
power.chisq.test(n=248, w=0.35, df=1, sig.level = 0.05, power = NULL) #
power = 0.9998087
power.chisq.test(n=248, w=0.4, df=1, sig.level = 0.05, power = NULL) #
power = 0.9999929
power.chisq.test(n=248, w=NULL, df=1, sig.level = 0.05, power = 0.5) # w =
0.1244504


# 1. Baseline GLM Combination

# Combining directly
GLM_1 <- glm(fake_s1$Timi23~fake_s1$Group, family = "binomial")
summary(GLM_1)
# Treatment effect: 0.6156(SE: 0.2778), p-value = 0.02667
```

```
confint(GLM_1)
#                   2.5 %    97.5 %
# (Intercept) 0.13517346 0.8628073
# fake_s1$Grp 0.07532953 1.1667334

fake_rej_glm1 <- function(Hist, Crnt, Histprob0, Histprob1, Crntprob0,
Crntprob1, n_sims=1000){
  signif <- rep(NA, n_sims)
  for (s in 1:n_sims) {
    fake_rej <- fake(Hist, Crnt, Histprob0, Histprob1, Crntprob0,
Crntprob1)
    glm_rej <- glm(Timi23 ~ Group, family = "binomial",data = fake_rej)
    beta_hat <- coef(glm_rej)["Group"]
    beta_se <- se.coef(glm_rej)["Group"]
    signif[s] <- ((beta_hat - 1.96*beta_se) > 0 | (beta_hat + 1.96*beta_se)
< 0)
    # Assume that we have double-tail hypotheses.
  }
  rejection <- mean(signif)
  return(rejection)
}
Grid <- c(seq(0.6, 0.9, by=0.05), 0.72)
Rslt_pwr_store_glm1 <- NULL
for (i in 1:length(Grid)) {
  str <- fake_rej_glm1(Hist = 21, Crnt = 228, Histprob0 = 0.7, Crntprob0 =
0.6, Histprob1 = 0.82, Crntprob1 = Grid[i], n_sims = 1000)
  Rslt_pwr_store_glm1 <- cbind(Rslt_pwr_store_glm1, str)
}
Rslt_pwr_store_glm1
# [1,] 0.047 0.136 0.386 0.705 0.923 0.993 0.999 0.516 | for 1000 updates

# 3. Two-stage Meta for fake_s1
Fake_twometa_01 <- twostage_lg_fit(data = fake_s1, trial_num = 2)
summary(Fake_twometa_01)
# coef: 0.6201(SE: 0.2790) | between-study heterogeneity(tau_beta): 0
# I-square stat = 1%
# It is significant(p = 0.0262)
confint(Fake_twometa_01)
#                   2.5 %   97.5 %
# (Intercept) 0.07326869 1.166864

fake_rej_twometa <- function(Hist, Crnt, Histprob0, Histprob1, Crntprob0,
Crntprob1, n_sims=1000){
  signif <- rep(NA, n_sims)
  for (s in 1:n_sims) {
    fake_rej <- fake(Hist, Crnt, Histprob0, Histprob1, Crntprob0,
Crntprob1)
    twometa_rej <- twostage_lg_fit(data = fake_rej, trial_num = 2)
    beta_hat <- coef(twometa_rej)
    beta_se <- sqrt(vcov(twometa_rej))
    signif[s] <- ((beta_hat - 1.96*beta_se) > 0 | (beta_hat + 1.96*beta_se)
< 0)
  }
  rejection <- mean(signif)
  return(rejection)
}

Rslt_pwr_store_twometa <- NULL
for (i in 1:length(Grid)) {
  str <- fake_rej_twometa(Hist = 21, Crnt = 228, Histprob0 = 0.7, Crntprob0
= 0.6, Histprob1 = 0.82, Crntprob1 = Grid[i], n_sims = 1000)
```

```
  Rslt_pwr_store_twometa <- cbind(Rslt_pwr_store_twometa, str)
}
Rslt_pwr_store_twometa
# [1,] 0.046 0.113 0.339 0.6 0.797 0.786 0.694 0.445 | for 1000 updates
# 1.96 [1,] 0.039 0.129 0.34 0.605 0.815 0.81 0.692 0.475

# 5. PQL
Fake_PQL <- glmmPQL(Timi23 ~ trialid + Group, random = ~Group+0|trialid,
family = binomial, data = fake_s1, verbose = FALSE, niter = 10)
summary(Fake_PQL)
# Treatment effect: 0.6240820 (SE:0.2807596)| between-study: 0.0001
# It is significant(p=0.0252)
fake_rej_PQL <- function(Hist, Crnt, Histprob0, Histprob1, Crntprob0,
Crntprob1, n_sims=1000){
  signif <- rep(NA, n_sims)
  for (s in 1:n_sims){
    fake_rej <- fake(Hist, Crnt, Histprob0, Histprob1, Crntprob0,
Crntprob1)
    PQL_rej <- glmmPQL(Timi23 ~ trialid + Group, random = ~Group+0|trialid,
family = binomial, data = fake_rej, verbose = FALSE, niter = 10)
    beta_hat <- fixef(PQL_rej)["Group"]
    beta_se <- sqrt(diag(as.matrix(vcov(PQL_rej)))["Group"])
    signif[s] <- ((beta_hat - 1.96*beta_se) > 0 | (beta_hat + 1.96*beta_se)
< 0)
  }
  rejection <- mean(signif)
  return(rejection)
}
Rslt_pwr_store_PQL <- NULL
for(i in 1:length(Grid)){
  str <- fake_rej_PQL(Hist = 21, Crnt = 228, Histprob0 = 0.7, Crntprob0 =
0.6, Histprob1 = 0.82, Crntprob1 = Grid[i], n_sims = 1000)
  Rslt_pwr_store_PQL <- cbind(Rslt_pwr_store_PQL, str)
}
Rslt_pwr_store_PQL
# [1,] 0.056 0.134 0.391 0.697 0.889 0.948 0.908 0.482
# 1.96 [1,] 0.048 0.155 0.416 0.718 0.902 0.949 NA 0.523


# 5. Standard Bayesian Hierarchical Modeling
set.seed(9999)
n_reps <- 200 # generating 500 times

btmean.Mu = 0
btprec.Mu = 0.01
thtmean.Mu = 0
thtprec.Mu = 0.01

# For power simulation
Hist_01 = 21
Crnt_01 = 228

Histprob_fk1_0 = 0.7
Histprob_fk1_1 = 0.82
Crntprob_fk1_0 = 0.6
# Crntprob_fk1_1 = seq(0.45,0.96,by=0.03)
# Crntprob_fk1_1 = seq(0.6, 0.8, by=0.05)
Crntprob_fk1_1 = c(0.7, 0.75)
# Pre-specify several borrowing rates.
bttau.alpha <-c(0,0,0);  bttau.beta<-c(0,0,0)
bttau.alpha[1] <- 5000;  bttau.beta[1] <- 50    # lots of borrowing
```

```
bttau.alpha[2] <- 300;  bttau.beta[2] <- 30     # a little borrowing
bttau.alpha[3] <- 10;  bttau.beta[3] <- 10      # not much borrowing

thttau.alpha <-c(0,0,0);  thttau.beta<-c(0,0,0)
thttau.alpha[1] <- 5000;  thttau.beta[1] <- 50    # lots of borrowing
thttau.alpha[2] <- 300;  thttau.beta[2] <- 30     # a little borrowing
thttau.alpha[3] <- 10;  thttau.beta[3] <- 10      # not much borrowing

dput(list(mubeta=1, prec=0.01, mutheta=1,
prec0=0.01),"fake_hier_inits.txt")

pwrbeta1_fk_hier_01 <- matrix(0,length(bttau.alpha),
length(Crntprob_fk1_1))
# pwrbeta1_fk_hier_01 <- rep(0,length(bttau.alpha))

# beta1store_fk_hier01 <- NULL;
# beta1stat_fk_hier_01 <- NULL; beta1id <- 0

loopstart <- Sys.time()
for (k in 1:length(bttau.alpha)) {
  for (j in 1:length(Crntprob_fk1_1)){
    rejectbeta1 <- 0;
    for (i in 1:n_reps) {
      bt1_new <- NULL
      fake_nnn <- fake(Hist = Hist_01, Crnt = Crnt_01, Histprob0 =
Histprob_fk1_0,
              Crntprob0 = Crntprob_fk1_0, Histprob1 = Histprob_fk1_1,
Crntprob1 = Crntprob_fk1_1[j])
      # fake_nnn <- fake(Hist = Hist_01, Crnt = Crnt_01, Histprob0 =
Histprob_fk1_0,
      #                  Crntprob0 = Crntprob_fk1_0, Histprob1 =
Histprob_fk1_1, Crntprob1 = Crntprob_fk1_1)
      x0_fake_01 <- fake_nnn$Group[fake_nnn$trialid == "1"]
      y0_fake_01 <- fake_nnn$Timi23[fake_nnn$trialid == "1"]
      x1_fake_01 <- fake_nnn$Group[fake_nnn$trialid == "2"]
      y1_fake_01 <- fake_nnn$Timi23[fake_nnn$trialid == "2"]

      dput(list(x0=as.numeric(x0_fake_01), y0=as.numeric(y0_fake_01),
                x1=as.numeric(x1_fake_01), y1=as.numeric(y1_fake_01),
                n0=Hist_01, n1=Crnt_01,
                btmean.Mu = btmean.Mu, btprec.Mu = btprec.Mu, thtmean.Mu =
thtmean.Mu,
                thtprec.Mu = thtprec.Mu, bttau.alpha = bttau.alpha[k],
bttau.beta = bttau.beta[k],
                thttau.alpha = thttau.alpha[k], thttau.beta =
thttau.beta[k]),"fake_hier_data.txt")

      modelCheck("BUGS_binary_hier.txt")
      modelData("fake_hier_data.txt")
      modelCompile(numChains=4)
      modelInits("fake_hier_inits.txt")
      modelGenInits()

      modelUpdate(3000)
      samplesSet(c("beta1"))
      modelUpdate(10000)

      # beta1id <- beta1id + 1
      # bt1_new <- cbind(samplesStats("beta1"), bttau.alpha[k],
bttau.beta[k], thttau.alpha, thttau.beta, beta1id)
      # beta1stat_fk_hier_01 <- rbind(beta1stat_fk_hier_01, bt1_new)
```

```
      # Decision rules
      LB <- samplesStats("beta1")$val2.5pc
      UB <- samplesStats("beta1")$val97.5pc
      if (LB > 0) rejectbeta1 <- rejectbeta1 + 1
      else
        if (UB < 0) rejectbeta1 <- rejectbeta1 + 1

    }
    pwrbeta1_fk_hier_01[k,j] <- rejectbeta1/n_reps
    # pwrbeta1_fk_hier_01[k] <- rejectbeta1/n_reps
    # cat("\nstats for beta_1:\n"); print(beta1stat_fk_hier_01)
    cat("\nPr(reject H0: beta_1=0) is: ", pwrbeta1_fk_hier_01[k,j])
    # cat("\nPr(reject H0: beta_1=0) is: ", pwrbeta1_fk_hier_01[k])
  }
}
tttime <- Sys.time() - loopstart
# 3000+10000: 14s
# two grids: 5h

# .6.65.7
#       [,1] [,2]  [,3]
# [1,] 0.040 0.17 0.365
# [2,] 0.035 0.12 0.320
# [3,] 0.050 0.08 0.335

# .75.8.85
#       [,1]  [,2]  [,3]
# [1,] 0.680 0.910 0.985
# [2,] 0.660 0.935 0.995
# [3,] 0.735 0.920 1.000

# 0.9
#       [,1]
# [1,] 1.000
# [2,] 1.000
# [3,] 0.995

# 0.72
#       [,1]
# [1,] 0.500
# [2,] 0.515
# [3,] 0.490


beta1stat_fk_hier_01

# Power plots for Binary Hierarchical Modeling
pwr_plt_prep01 <- pwrbeta1_fk_hier_01_store.45_.95
pwr_plt_prep01 <- rbind(t(pwr_plt_prep01[1,,drop=FALSE]),
t(pwr_plt_prep01[2,,drop=FALSE]), t(pwr_plt_prep01[3,,drop=FALSE]))
Paras <- rep(seq(0.45,0.95,by = 0.05), 3)
Prior.No <- as.character(c(rep("Not much Borrowing",11),rep("Moderate
Borrowing",11),rep("Lots of Borrowing",11)))
pwr_plt_prep01 <- cbind(pwr_plt_prep01, Paras, Prior.No)
colnames(pwr_plt_prep01)[1] <- "Power"

pwr_plt_prep01 <- as.data.frame(pwr_plt_prep01)
pwr_bin_hier_plt <- ggplot(data = pwr_plt_prep01, aes(x = Paras, y = Power,
group = Prior.No)) +
  geom_point(aes(color=Prior.No, shape=Prior.No)) + theme_bw() +
```

```
  scale_y_discrete(breaks=seq(0,1,by=0.05)) +
  theme(plot.title=element_text(hjust=0.5)) +
  ggtitle(label = "The Power plot of Std. BHM") +
  labs(x="True response rate", y="Power")

pwr_bin_hier_plt

# plot(lowess(Crntprob_fk1_1, pwrbeta1_fk_hier_01_store.45_.
95[1,],f=1),type="l",ylab="power",ylim=c(0,1),
#      xlab="true response rate",cex=1.5)
#
# lines(lowess(Crntprob_fk1_1, pwrbeta1_fk_hier_01_store.45_.
95[2,],f=2),lty=1)
# lines(lowess(Crntprob_fk1_1, pwrbeta1_fk_hier_01_store.45_.
95[3,],f=2),lty=1)

# 6. Unnormalized Power Prior
set.seed(9999)
rstan_options(auto_write = TRUE)
options(mc.cores = 4)
P <- 1 # Only the difference between Historical or Current

# Initialization
as <- .5
bs <- 2
vb <- 1.5

eta <- 1
nu <- 1

fake_PP_01 <- fake_s1

N_Hist_PP_fk <- 21
X_Hist_PP_fk <- matrix(NA, nrow = N_Hist_PP_fk, ncol = P)
X_Hist_PP_fk <- fake_PP_01$Group[fake_PP_01$trialid == "1"]
Y_Hist_PP_fk <- fake_PP_01$Timi23[fake_PP_01$trialid == "1"]
X_Hist_PP_fk <- as.matrix(X_Hist_PP_fk)


### will draw from the same process for simplicity
N_Cur_PP_fk <- 228
X_Cur_PP_fk <- matrix(NA, nrow = N_Cur_PP_fk, ncol = P)
X_Cur_PP_fk <- fake_PP_01$Group[fake_PP_01$trialid == "2"]
Y_Cur_PP_fk <- fake_PP_01$Timi23[fake_PP_01$trialid == "2"]
X_Cur_PP_fk <- as.matrix(X_Cur_PP_fk)

summary(glm(Y_Cur_PP_fk ~ X_Cur_PP_fk, family = "binomial"))
# Treatment effect: 0.5453(SE: 0.2923), p-value = 0.06211 (Exactly the same
with GLM_2)


lgr_fk_data <- list(
  N0 = N_Hist_PP_fk,
  X0 = X_Hist_PP_fk,
  P = P,
  y0 = Y_Hist_PP_fk,
  eta = eta,
  nu = nu,
  X = X_Cur_PP_fk,
  N = N_Cur_PP_fk,
  y = Y_Cur_PP_fk
```

```
)



Unnormalized_PP_rslt_fk01 <- Unnormalized_PP_posterior(model =
"Unnormal_PP.stan", data = lgr_fk_data, iter=13000, warmup = 3000)
#                 mean      se_mean         sd          2.5%           25%
50%          75%        97.5%    n_eff     Rhat
# alpha      0.5719185 0.0013721597 0.18753095  2.053876e-01    0.44615822
0.57163670    0.6962755    0.9438842 18678.27 1.000075
# beta[1]    0.5347905 0.0020619661 0.27638274 -7.168421e-03    0.34893432
0.53507778    0.7217206    1.0726501 17966.31 1.000056
# a_0        0.0687533 0.0004172342 0.06757031  1.803890e-03    0.02015958
0.04845613    0.0955325    0.2488220 26227.21 1.000219
# lp__     -145.4630020 0.0112806715 1.26302436 -1.487058e+02 -146.05933310
-145.13350522 -144.5392163 -144.0090692 12535.84 1.000240

Reject_Unnormalized_PP <- function(Hist, Crnt, Histprob0, Histprob1,
Crntprob0, Crntprob1, stancode, P, nu, eta, n_sims = 500){
  rej <- 0
  for (i in 1:n_sims) {
      fake_PP <- fake(Hist, Crnt, Histprob0, Histprob1, Crntprob0,
Crntprob1)
      N_Hist_PP <- Hist
      X_Hist_PP <- matrix(NA, nrow = N_Hist_PP, ncol = P)
      X_Hist_PP <- fake_PP$Group[fake_PP$trialid == "1"]
      Y_Hist_PP <- fake_PP$Timi23[fake_PP$trialid == "1"]
      X_Hist_PP <- as.matrix(X_Hist_PP)

      ### will draw from the same process for simplicity
      N_Cur_PP <- Crnt
      X_Cur_PP <- matrix(NA, nrow = N_Cur_PP, ncol = P)
      X_Cur_PP <- fake_PP$Group[fake_PP$trialid == "2"]
      Y_Cur_PP <- fake_PP$Timi23[fake_PP$trialid == "2"]
      X_Cur_PP <- as.matrix(X_Cur_PP)

      lgr_fk_data <- list(
        N0 = N_Hist_PP,
        X0 = X_Hist_PP,
        P = P,
        y0 = Y_Hist_PP,
        eta = eta,
        nu = nu,
        X = X_Cur_PP,
        N = N_Cur_PP,
        y = Y_Cur_PP
      )
      file.remove("Unnormal_PP.rds")
      Unnormalized_PP_rslt_fk <- Unnormalized_PP_posterior(model =
stancode, data = lgr_fk_data, iter = 13000, warmup = 3000)
      LB <- Unnormalized_PP_rslt_fk$summary[2,4]
      UB <- Unnormalized_PP_rslt_fk$summary[2,8]
      if(LB > 0) rej = rej + 1
      else
        if(UB < 0) rej = rej + 1
      file.remove("Unnormal_PP.rds")
  }
  rej_rate <- rej/n_sims
      return(rej_rate)
}
```

```
strt <- Sys.time()
Pwr_Unm_PP_store <- NULL
for (i in 1:length(Grid)) {
  str <- Reject_Unnormalized_PP(Hist = 21, Crnt = 228, Histprob0 = 0.7,
Histprob1 = 0.82,
                                Crntprob0 = 0.6, Crntprob1 = Grid[i],
n_sims = 5,
                                stancode = "Unnormal_PP.stan",
                                P = 1, nu = 1, eta = 1)# n_sims = 10 ->
1min
  Pwr_Unm_PP_store <- cbind(Pwr_Unm_PP_store, str)
  cat('grid_num:',i,'in',length(Grid),'\n')
}
tm <- Sys.time()-strt
Pwr_Unm_PP_store

# 0.6.65
# [1,] 0.1 0.095
# 0.7.75
# [1,] 0.41 0.66
# 0.8.85
# [1,] 0.885 0.975
# 0.9.72
# [1,] 0.995 0.48


# 7. Normalized Power Prior
Reject_NPP <- function(Hist, Crnt, Histprob0, Histprob1, Crntprob0,
Crntprob1, prr_stancode, postrr_stancode, P, nu, eta, n_sims = 500){
  rej <- 0
  for (i in 1:n_sims) {
    fake_PP <- fake(Hist, Crnt, Histprob0, Histprob1, Crntprob0, Crntprob1)
    N_Hist_PP <- Hist
    X_Hist_PP <- matrix(NA, nrow = N_Hist_PP, ncol = P)
    X_Hist_PP <- fake_PP$Group[fake_PP$trialid == "1"]
    Y_Hist_PP <- fake_PP$Timi23[fake_PP$trialid == "1"]
    X_Hist_PP <- as.matrix(X_Hist_PP)

    ### will draw from the same process for simplicity
    N_Cur_PP <- Crnt
    X_Cur_PP <- matrix(NA, nrow = N_Cur_PP, ncol = P)
    X_Cur_PP <- fake_PP$Group[fake_PP$trialid == "2"]
    Y_Cur_PP <- fake_PP$Timi23[fake_PP$trialid == "2"]
    X_Cur_PP <- as.matrix(X_Cur_PP)

    lgr_fk_data <- list(
      N0 = N_Hist_PP,
      X0 = X_Hist_PP,
      P = P,
      y0 = Y_Hist_PP,
      eta = eta,
      nu = nu,
      X = X_Cur_PP,
      N = N_Cur_PP,
      y = Y_Cur_PP
    )
    lgr_fk_fora0_data <- list(
      N0 = N_Hist_PP,
      X0 = X_Hist_PP,
      P = P,
      y0 = Y_Hist_PP,
```

```
      a_0 = NULL
    )
    cat('data generated\n')
    Approximate_NPP_rslt_fk <- Approximation_Normalized_PP(prior_stancode =
prr_stancode,
                                                           postrr_stancode
= postrr_stancode,
                                                           data_postrr =
lgr_fk_data,
                                                           data_a0 =
lgr_fk_fora0_data)
    LB <- Approximate_NPP_rslt_fk$summary[2,4]
    UB <- Approximate_NPP_rslt_fk$summary[2,8]
    if(LB > 0) rej = rej + 1
    else
      if(UB < 0) rej = rej + 1

    cat('loops = ', i, 'in', n_sims, '\n', 'loop_time =', Sys.time()-strt,
'\n')
  }
  rej_rate <- rej/n_sims
  return(rej_rate)
}

strt <- Sys.time()
Pwr_NPP_store <- NULL
for (i in 1:length(Grid)) {
  str <- Reject_NPP(Hist = 21, Crnt = 228, Histprob0 = 0.7, Histprob1 =
0.82,
                    Crntprob0 = 0.6, Crntprob1 = Grid[i], n_sims = 5,
                    prr_stancode = "logistic_regression_prior.stan",
                    postrr_stancode = "NPP_approximate.stan",
                    P = 1, nu = 1, eta = 1)# n_sims = 5 -> ..min
  Pwr_NPP_store <- cbind(Pwr_NPP_store, str)
  cat('grid_num:',i,'\n')
}
tm <- Sys.time()-strt
Pwr_NPP_store

# Grid=0.8 [1,] 0.925
# 0.75 [1,] 0.715
# 0.7 [1,] 0.405
# 0.65 [1,] 0.14
# 0.6 [1,] 0.02
# 0.9 [1,]   1
# 0.85 [1,] 0.98
# 0.72 [1,] 0.49

# 8. Commensurate Prior

set.seed(9999)
n_reps <- 200 # generating 500 times

# For power simulation
Hist_01 = 21
Crnt_01 = 228

Histprob_fk1_0 = 0.7
Histprob_fk1_1 = 0.82
Crntprob_fk1_0 = 0.6
# Crntprob_fk1_1 = seq(0.45,0.96,by=0.03)
```

```
# Crntprob_fk1_1 = seq(0.6, 0.8, by=0.05)
Crntprob_fk1_1 = c(seq(0.6, 0.9, by=0.05), 0.72)

dput(list(beta0=0, theta0=0),"fake_CP_inits.txt")

pwrbeta1_fk_CP <- rep(0, length(Crntprob_fk1_1))
# pwrbeta1_fk_hier_01 <- rep(0,length(bttau.alpha))

loopstart <- Sys.time()
for (j in 1:length(Crntprob_fk1_1)){
  rejectbeta1 <- 0;
  for (i in 1:n_reps) {
    bt1_new <- NULL
    fake_nnn <- fake(Hist = Hist_01, Crnt = Crnt_01, Histprob0 =
Histprob_fk1_0,
                     Crntprob0 = Crntprob_fk1_0, Histprob1 =
Histprob_fk1_1, Crntprob1 = Crntprob_fk1_1[j])
    # fake_nnn <- fake(Hist = Hist_01, Crnt = Crnt_01, Histprob0 =
Histprob_fk1_0,
    #                  Crntprob0 = Crntprob_fk1_0, Histprob1 =
Histprob_fk1_1, Crntprob1 = Crntprob_fk1_1)
    x0_fake_01 <- fake_nnn$Group[fake_nnn$trialid == "1"]
    y0_fake_01 <- fake_nnn$Timi23[fake_nnn$trialid == "1"]
    x1_fake_01 <- fake_nnn$Group[fake_nnn$trialid == "2"]
    y1_fake_01 <- fake_nnn$Timi23[fake_nnn$trialid == "2"]

    dput(list(x0=as.numeric(x0_fake_01), y0=as.numeric(y0_fake_01),
              x1=as.numeric(x1_fake_01), y1=as.numeric(y1_fake_01),
              n0=Hist_01, n1=Crnt_01, spike= 200, p.choose=0.9,
              l.slab=0.01, u.slab=100), "fake_CP_data.txt")

    modelCheck("CP_BUGS_wout_cov.txt")
    modelData("fake_CP_data.txt")
    modelCompile(numChains=4)
    modelInits("fake_CP_inits.txt")
    modelGenInits()

    modelUpdate(3000)
    samplesSet(c("beta1"))
    modelUpdate(10000)

    # Decision rules
    LB <- samplesStats("beta1")$val2.5pc
    UB <- samplesStats("beta1")$val97.5pc
    if (LB > 0) rejectbeta1 <- rejectbeta1 + 1
    else
      if (UB < 0) rejectbeta1 <- rejectbeta1 + 1
    cat("loop = ",i,"in",n_reps,"\n Grid = ",j,"in",length(Crntprob_fk1_1))
    }
    pwrbeta1_fk_CP[j] <- rejectbeta1/n_reps
    cat("\nPr(reject H0: beta_1=0) is: ", pwrbeta1_fk_CP[j])
  }
tttime <- Sys.time() - loopstart # 2grids * 2loops = 1.3min

# [1] 0.045 0.175 0.395 0.685 0.920 0.995 1.000 0.485


# 9. Commensurate Power Prior
Reject_CPP <- function(Hist, Crnt, Histprob0, Histprob1, Crntprob0,
Crntprob1, stancode, P, n_sims = 500){
  rej <- 0
```

```
  for (i in 1:n_sims) {
    fake_PP <- fake(Hist, Crnt, Histprob0, Histprob1, Crntprob0, Crntprob1)
    N_Hist_PP <- Hist
    X_Hist_PP <- matrix(NA, nrow = N_Hist_PP, ncol = P)
    X_Hist_PP <- fake_PP$Group[fake_PP$trialid == "1"]
    Y_Hist_PP <- fake_PP$Timi23[fake_PP$trialid == "1"]
    X_Hist_PP <- as.matrix(X_Hist_PP)

    ### will draw from the same process for simplicity
    N_Cur_PP <- Crnt
    X_Cur_PP <- matrix(NA, nrow = N_Cur_PP, ncol = P)
    X_Cur_PP <- fake_PP$Group[fake_PP$trialid == "2"]
    Y_Cur_PP <- fake_PP$Timi23[fake_PP$trialid == "2"]
    X_Cur_PP <- as.matrix(X_Cur_PP)

    lgr_fk_data <- list(
      N0 = N_Hist_PP,
      X0 = X_Hist_PP,
      P = P,
      y0 = Y_Hist_PP,
      X1 = X_Cur_PP,
      N1 = N_Cur_PP,
      y1 = Y_Cur_PP
    )
    cat('data generated\n')
    CPP_rslt_fk <- CPP_get_posterior(stancode = stancode, data =
lgr_fk_data, iter = 13000, warmup = 3000)
    LB <- CPP_rslt_fk$summary[4,4]
    UB <- CPP_rslt_fk$summary[4,8]
    if(LB > 0) rej = rej + 1
    else
      if(UB < 0) rej = rej + 1
    # file.remove("Unnormal_PP.rds")
    cat('loops = ', i, 'in', n_sims, '\n', 'loop_time =', Sys.time()-strt,
'\n')
  }
  rej_rate <- rej/n_sims
  return(rej_rate)
}

strt <- Sys.time()
Pwr_CPP_store <- NULL
for (i in 1:length(Grid)) {
  str <- Reject_CPP(Hist = 21, Crnt = 228, Histprob0 = 0.7, Histprob1 =
0.82,
                    Crntprob0 = 0.6, Crntprob1 = Grid[i], n_sims = 200,
                    stancode = "CPP_wout_cov.stan", P = 1)# n_sims = 10 ->
1min
  Pwr_CPP_store <- cbind(Pwr_CPP_store, str)
  cat('grid_num:',i,'in',length(Grid),'\n')
}
tm <- Sys.time()-strt
Pwr_CPP_store
# 0.6.65
# [1,] 0.03 0.115
# 0.7.75
# [1,] 0.345 0.68
# 0.8.85
# [1,] 0.945 0.995
# 0.9.72
# [1,]   1 0.495
```

```
######### Fake-data Type-I error ####################################

# 1. Baseline GLM Type-I error
GLM_1_TI <- glm(fake_s2$Timi23 ~ fake_s2$Group, family = "binomial")
summary(GLM_1_TI)
# Treatment Effect: 0.3888(SE: 0.2623), p-value = 0.138
confint(GLM_1_TI)
#                     2.5 %    97.5 %
# (Intercept)   -0.06127343 0.6522033
# fake_s2$Group -0.12372083 0.9062275

store_fk_TI_glm1 <- fake_rej_glm1(Hist = 21, Crnt = 228, Histprob0 = 0.7,
Crntprob0 = 0.6, Histprob1 = 0.82, Crntprob1 = 0.6, n_sims = 1000) #
Histprob0 <-> Histprob1; Crntprob0 <-> Crntprob1
store_fk_TI_glm1 # Type-I error = 0.0462 for 5000; 0.053 for 1000 updates

# 2. Two-stage Type-I error
Fake_twometa_02 <- twostage_lg_fit(data = fake_s2, trial_num = 2)
summary(Fake_twometa_02)
# Treatment Effect: 0.2921(SE: 0.2708), p-value = 0.2807
confint(Fake_twometa_02)
#                   2.5 %    97.5 %
# (Intercept) -0.2385883 0.8228614

store_fk_TI_twometa <- fake_rej_twometa(Hist = 21, Crnt = 228, Histprob0 =
0.7, Crntprob0 = 0.6, Histprob1 = 0.82, Crntprob1 = 0.6, n_sims = 1000)
store_fk_TI_twometa # power = 0.043

# 3. One-stage Type-I error
Fake_onemeta_02 <- glmer(Timi23 ~ trialid + Group + (Group+0|trialid),
family = "binomial", data = fake_s2)
summary(Fake_onemeta_02)
# Treatment Effect: 0.3893(SE: 0.2653), p-value = 0.1422
confint(Fake_onemeta_02)
#                   2.5 %    97.5 %
# .sig01       0.0000000 5.3279909
# (Intercept)  0.3815315 5.9261850
# trialid     -2.8749102 0.0120254
# Group       -0.1539761 5.2476751


# store_fk_TI_onemeta <- fake_rej_onemeta(Hist = 21, Crnt = 228, Histprob0
= 0.7, Crntprob0 = 0.6, Histprob1 = 0.82, Crntprob1 = 0.6, n_sims = 1000)
# store_fk_TI_onemeta

# 4. Std. BHM Type-I error(Similar to Power analysis)
set.seed(9999)
n_reps <- 500 # generating 500 times

btmean.Mu = 0
btprec.Mu = 0.1
thtmean.Mu = 0
thtprec.Mu = 0.1

# For power simulation
Hist_02 = 21
Crnt_02 = 228

Histprob_fk2_0 = 0.7
Histprob_fk2_1 = 0.82
```

```
Crntprob_fk2_0 = 0.6
Crntprob_fk2_1 = 0.6

# Pre-specify several borrowing rates.
bttau.alpha <-c(0,0,0);  bttau.beta<-c(0,0,0)
bttau.alpha[1] <- 2;  bttau.beta[1] <- 2    # lots of borrowing
bttau.alpha[2] <- 6;  bttau.beta[2] <- 6    # a little borrowing
bttau.alpha[3] <- 16;  bttau.beta[3] <- 16  # not much borrowing

thttau.alpha <- 2;  thttau.beta <- 2    # lots of borrowing

dput(list(mubeta=1, prec=0.1, mutheta=1, prec0=0.1),"fake_hier_inits.txt")

# TI_beta1_fk_hier_02 <- matrix(0,length(bttau.alpha),
length(Crntprob_fk2_1))
TI_beta1_fk_hier_02 <- rep(0,length(bttau.alpha))

# beta1store_fk_hier02 <- NULL;
beta1stat_fk_hier_02 <- NULL; beta1id <- 0

loopstart <- Sys.time()
for (k in 1:length(bttau.alpha)) {
    rejectbeta1 <- 0;
    for (i in 1:n_reps) {
      bt1_new <- NULL
      fake_nnn <- fake(Hist = Hist_02, Crnt = Crnt_02, Histprob0 =
Histprob_fk2_0,
                       Crntprob0 = Crntprob_fk2_0, Histprob1 =
Histprob_fk2_1, Crntprob1 = Crntprob_fk2_1)
      x0_fake_02 <- fake_nnn$Group[fake_nnn$trialid == "1"]
      y0_fake_02 <- fake_nnn$Timi23[fake_nnn$trialid == "1"]
      x1_fake_02 <- fake_nnn$Group[fake_nnn$trialid == "2"]
      y1_fake_02 <- fake_nnn$Timi23[fake_nnn$trialid == "2"]

      dput(list(x0=as.numeric(x0_fake_02), y0=as.numeric(y0_fake_02),
                x1=as.numeric(x1_fake_02), y1=as.numeric(y1_fake_02),
                n0=Hist_02, n1=Crnt_02,
                btmean.Mu = btmean.Mu, btprec.Mu = btprec.Mu, thtmean.Mu =
thtmean.Mu,
                thtprec.Mu = thtprec.Mu, bttau.alpha = bttau.alpha[k],
bttau.beta = bttau.beta[k],
                thttau.alpha = thttau.alpha, thttau.beta =
thttau.beta),"fake_hier_data.txt")

      modelCheck("BUGS_binary_hier.txt")
      modelData("fake_hier_data.txt")
      modelCompile(numChains=1)
      modelInits("fake_hier_inits.txt")
      modelGenInits()

      modelUpdate(2000)
      samplesSet(c("beta1"))
      modelUpdate(5000)

      # beta1id <- beta1id + 1
      # bt1_new <- cbind(samplesStats("beta1"), bttau.alpha[k],
bttau.beta[k], thttau.alpha, thttau.beta, beta1id)
      # beta1stat_fk_hier_02 <- rbind(beta1stat_fk_hier_02, bt1_new)

      # Decision rules
      LB <- samplesStats("beta1")$val2.5pc
```

```
      UB <- samplesStats("beta1")$val97.5pc
      if (LB > 0) rejectbeta1 <- rejectbeta1 + 1
      else
        if (UB < 0) rejectbeta1 <- rejectbeta1 + 1

    }
    TI_beta1_fk_hier_02[k] <- rejectbeta1/n_reps
    # cat("\nstats for beta_1:\n"); print(beta1stat_fk_hier_02)
    cat("\nPr(reject H0: beta_1=0 when it is true) is: ",
TI_beta1_fk_hier_02[k])
}
tttime <- Sys.time() - loopstart
TI_beta1_fk_hier_02_store <- TI_beta1_fk_hier_02
TI_beta1_fk_hier_02_store
# [1] 0.050 0.050 0.048

# 5. Unnormalized Power Prior
TI_Unm_PP_store <- Reject_Unnormalized_PP(Hist = 21, Crnt = 228, Histprob0
= 0.7, Histprob1 = 0.82,
                                          Crntprob0 = 0.6, Crntprob1 =
0.6, n_sims = 500,
                                          stancode =
"simple_logistic_regression_posterior_unnormalised.stan",
                                          P = 1, nu = 1, eta = 1)# n_sims
= 10 -> 1min
TI_Unm_PP_store
# [1] 0.054


######### Simulation 2 equal sample size 0.3 0.7 ############
# Two meta

Rslt_pwr_store_twometa_02 <- NULL
for (i in 1:length(Grid)) {
  str <- fake_rej_twometa(Hist = 228, Crnt = 228, Histprob0 = 0.7,
Crntprob0 = 0.6, Histprob1 = 0.82, Crntprob1 = Grid[i], n_sims = 1000)
  Rslt_pwr_store_twometa_02 <- cbind(Rslt_pwr_store_twometa_02, str)
}
Rslt_pwr_store_twometa_02
# [1,] 0.096 0.348 0.607 0.82 0.847 0.742 0.589 0.695

# PQL

Rslt_pwr_store_PQL_02 <- NULL
for(i in 1:length(Grid)){
  str <- fake_rej_PQL(Hist = 228, Crnt = 228, Histprob0 = 0.7, Crntprob0 =
0.6, Histprob1 = 0.82, Crntprob1 = Grid[i], n_sims = 1000)
  Rslt_pwr_store_PQL_02 <- cbind(Rslt_pwr_store_PQL_02, str)
}
Rslt_pwr_store_PQL_02
# [1,] 0.21 0.463 0.72 0.875 0.95 0.943 0.9 0.769



# Commensurate Prior
# simulation2
set.seed(9999)
n_reps <- 200 # generating 500 times

# For power simulation
Hist_01 = 228 # same sample size
```

```
Crnt_01 = 228

Histprob_fk1_0 = 0.7
Histprob_fk1_1 = 0.82
Crntprob_fk1_0 = 0.6
# Crntprob_fk1_1 = seq(0.45,0.96,by=0.03)
# Crntprob_fk1_1 = seq(0.6, 0.8, by=0.05)
# Crntprob_fk1_1 = c(seq(0.6, 0.9, by=0.05), 0.72)
Crntprob_fk1_1 = c(0.7,0.75)
dput(list(beta0=0, theta0=0),"fake_CP_inits.txt")

pwrbeta1_fk_CP_02 <- rep(0, length(Crntprob_fk1_1))
# pwrbeta1_fk_hier_01 <- rep(0,length(bttau.alpha))

loopstart <- Sys.time()
for (j in 1:length(Crntprob_fk1_1)){
  rejectbeta1 <- 0;
  for (i in 1:n_reps) {
    bt1_new <- NULL
    fake_nnn <- fake(Hist = Hist_01, Crnt = Crnt_01, Histprob0 =
Histprob_fk1_0,
                     Crntprob0 = Crntprob_fk1_0, Histprob1 =
Histprob_fk1_1, Crntprob1 = Crntprob_fk1_1[j])
    # fake_nnn <- fake(Hist = Hist_01, Crnt = Crnt_01, Histprob0 =
Histprob_fk1_0,
    #                  Crntprob0 = Crntprob_fk1_0, Histprob1 =
Histprob_fk1_1, Crntprob1 = Crntprob_fk1_1)
    x0_fake_01 <- fake_nnn$Group[fake_nnn$trialid == "1"]
    y0_fake_01 <- fake_nnn$Timi23[fake_nnn$trialid == "1"]
    x1_fake_01 <- fake_nnn$Group[fake_nnn$trialid == "2"]
    y1_fake_01 <- fake_nnn$Timi23[fake_nnn$trialid == "2"]

    dput(list(x0=as.numeric(x0_fake_01), y0=as.numeric(y0_fake_01),
              x1=as.numeric(x1_fake_01), y1=as.numeric(y1_fake_01),
              n0=Hist_01, n1=Crnt_01, spike= 200, p.choose=0.9,
              l.slab=0.01, u.slab=100), "fake_CP_data.txt")

    modelCheck("CP_BUGS_wout_cov.txt")
    modelData("fake_CP_data.txt")
    modelCompile(numChains=4)
    modelInits("fake_CP_inits.txt")
    modelGenInits()

    modelUpdate(3000)
    samplesSet(c("beta1"))
    modelUpdate(10000)

    # Decision rules
    LB <- samplesStats("beta1")$val2.5pc
    UB <- samplesStats("beta1")$val97.5pc
    if (LB > 0) rejectbeta1 <- rejectbeta1 + 1
    else
      if (UB < 0) rejectbeta1 <- rejectbeta1 + 1
    cat("loop = ",i,"in",n_reps,"\n Grid = ",j,"in",length(Crntprob_fk1_1))
  }
  pwrbeta1_fk_CP_02[j] <- rejectbeta1/n_reps
  cat("\nPr(reject H0: beta_1=0) is: ", pwrbeta1_fk_CP_02[j])
}
tttime <- Sys.time() - loopstart
# 0.7.75 [1] 0.585 0.850
```

```
# CPP
strt <- Sys.time()
Pwr_CPP_store_02 <- NULL
for (i in 1:length(Grid)) {
  str <- Reject_CPP(Hist = 228, Crnt = 228, Histprob0 = 0.7, Histprob1 =
0.82,
                    Crntprob0 = 0.6, Crntprob1 = Grid[i], n_sims = 200,
                    stancode = "CPP_wout_cov.stan", P = 1)# n_sims = 10 ->
1min
  Pwr_CPP_store_02 <- cbind(Pwr_CPP_store_02, str)
  cat('grid_num:',i,'in',length(Grid),'\n')
}
tm <- Sys.time()-strt
Pwr_CPP_store_02
# 0.6.65
# [1,] 0.59 0.58
# 0.7.75
# [1,] 0.515 0.535


######### Scenario 1: Imbl 0.6 0.72 0.6 Grids
########################################

# GLM
Grid <- c(seq(0.6, 0.9, by=0.05), 0.72)
Rslt_pwr_store_glm_Sc1 <- NULL
for (i in 1:length(Grid)) {
  str <- fake_rej_glm1(Hist = 21, Crnt = 228, Histprob0 = 0.6, Crntprob0 =
0.6, Histprob1 = 0.72, Crntprob1 = Grid[i], n_sims = 1000)
  Rslt_pwr_store_glm_Sc1 <- cbind(Rslt_pwr_store_glm_Sc1, str)
}
Rslt_pwr_store_glm_Sc1
# [1,] 0.057 0.139 0.394 0.674 0.896 0.986   1 0.491


Grid <- c(seq(0.6,0.9,by=0.05),0.72)
# Two meta
Rslt_pwr_store_twometa_Sc1 <- NULL
for (i in 1:length(Grid)) {
  str <- fake_rej_twometa(Hist = 228, Crnt = 228, Histprob0 = 0.6,
Crntprob0 = 0.6, Histprob1 = 0.72, Crntprob1 = Grid[i], n_sims = 1000)
  Rslt_pwr_store_twometa_Sc1 <- cbind(Rslt_pwr_store_twometa_Sc1, str)
}
Rslt_pwr_store_twometa_Sc1
# [1,] 0.112 0.326 0.576 0.714 0.732 0.607 0.394 0.659

# PQL
Grid <- c(seq(0.75,0.9,by=0.05),0.72)
Rslt_pwr_store_PQL_Sc1 <- NULL
for(i in 1:length(Grid)){
  str <- fake_rej_PQL(Hist = 21, Crnt = 228, Histprob0 = 0.6, Crntprob0 =
0.6, Histprob1 = 0.72, Crntprob1 = Grid[i], n_sims = 10000)
  Rslt_pwr_store_PQL_Sc1 <- cbind(Rslt_pwr_store_PQL_Sc1, str)
}
Rslt_pwr_store_PQL_Sc1
# [1,] 0.051 0.144 0.39 0.69 0.87 0.936 0.886 0.515

# 5. Standard Bayesian Hierarchical Modeling
set.seed(9999)
n_reps <- 200 # generating 500 times
```

```
btmean.Mu = 0
btprec.Mu = 0.01
thtmean.Mu = 0
thtprec.Mu = 0.01

# For power simulation
Hist_01 = 21
Crnt_01 = 228

Histprob_fk1_0 = 0.6
Histprob_fk1_1 = 0.72
Crntprob_fk1_0 = 0.6
# Crntprob_fk1_1 = seq(0.45,0.96,by=0.03)
# Crntprob_fk1_1 = seq(0.6, 0.8, by=0.05)
Crntprob_fk1_1 = c(0.85, 0.9,0.72)
# Pre-specify several borrowing rates.
bttau.alpha <-c(0,0,0);  bttau.beta<-c(0,0,0)
bttau.alpha[1] <- 5000;  bttau.beta[1] <- 50    # lots of borrowing
bttau.alpha[2] <- 300;  bttau.beta[2] <- 30     # a little borrowing
bttau.alpha[3] <- 10;  bttau.beta[3] <- 10      # not much borrowing

thttau.alpha <-c(0,0,0);  thttau.beta<-c(0,0,0)
thttau.alpha[1] <- 5000;  thttau.beta[1] <- 50    # lots of borrowing
thttau.alpha[2] <- 300;  thttau.beta[2] <- 30     # a little borrowing
thttau.alpha[3] <- 10;  thttau.beta[3] <- 10      # not much borrowing

dput(list(mubeta=1, prec=0.01, mutheta=1,
prec0=0.01),"fake_hier_inits.txt")

pwrbeta1_fk_hier_Sc1 <- matrix(0,length(bttau.alpha),
length(Crntprob_fk1_1))
# pwrbeta1_fk_hier_01 <- rep(0,length(bttau.alpha))

# beta1store_fk_hier01 <- NULL;
# beta1stat_fk_hier_01 <- NULL; beta1id <- 0

loopstart <- Sys.time()
# for (k in 1:length(bttau.alpha)) {
  # for (j in 1:length(Crntprob_fk1_1)){
    rejectbeta1 <- 0;k=3;j=3
    for (i in 1:n_reps) {
      bt1_new <- NULL
      fake_nnn <- fake(Hist = Hist_01, Crnt = Crnt_01, Histprob0 =
Histprob_fk1_0,
                       Crntprob0 = Crntprob_fk1_0, Histprob1 =
Histprob_fk1_1, Crntprob1 = Crntprob_fk1_1[j])
      # fake_nnn <- fake(Hist = Hist_01, Crnt = Crnt_01, Histprob0 =
Histprob_fk1_0,
      #                  Crntprob0 = Crntprob_fk1_0, Histprob1 =
Histprob_fk1_1, Crntprob1 = Crntprob_fk1_1)
      x0_fake_01 <- fake_nnn$Group[fake_nnn$trialid == "1"]
      y0_fake_01 <- fake_nnn$Timi23[fake_nnn$trialid == "1"]
      x1_fake_01 <- fake_nnn$Group[fake_nnn$trialid == "2"]
      y1_fake_01 <- fake_nnn$Timi23[fake_nnn$trialid == "2"]

      dput(list(x0=as.numeric(x0_fake_01), y0=as.numeric(y0_fake_01),
                x1=as.numeric(x1_fake_01), y1=as.numeric(y1_fake_01),
                n0=Hist_01, n1=Crnt_01,
                btmean.Mu = btmean.Mu, btprec.Mu = btprec.Mu, thtmean.Mu =
thtmean.Mu,
```

```
                thtprec.Mu = thtprec.Mu, bttau.alpha = bttau.alpha[k],
bttau.beta = bttau.beta[k],
                thttau.alpha = thttau.alpha[k], thttau.beta =
thttau.beta[k]),"fake_hier_data.txt")

      modelCheck("BUGS_binary_hier.txt")
      modelData("fake_hier_data.txt")
      modelCompile(numChains=4)
      modelInits("fake_hier_inits.txt")
      modelGenInits()

      modelUpdate(3000)
      samplesSet(c("beta1"))
      modelUpdate(10000)

      # beta1id <- beta1id + 1
      # bt1_new <- cbind(samplesStats("beta1"), bttau.alpha[k],
bttau.beta[k], thttau.alpha, thttau.beta, beta1id)
      # beta1stat_fk_hier_01 <- rbind(beta1stat_fk_hier_01, bt1_new)

      # Decision rules
      LB <- samplesStats("beta1")$val2.5pc
      UB <- samplesStats("beta1")$val97.5pc
      if (LB > 0) rejectbeta1 <- rejectbeta1 + 1
      else
        if (UB < 0) rejectbeta1 <- rejectbeta1 + 1

    }
    pwrbeta1_fk_hier_Sc1[k,j] <- rejectbeta1/n_reps
    # pwrbeta1_fk_hier_01[k] <- rejectbeta1/n_reps
    # cat("\nstats for beta_1:\n"); print(beta1stat_fk_hier_01)
    cat("\nPr(reject H0: beta_1=0) is: ", pwrbeta1_fk_hier_Sc1[k,j])
    # cat("\nPr(reject H0: beta_1=0) is: ", pwrbeta1_fk_hier_01[k])
#   }
# }
tttime <- Sys.time() - loopstart
pwrbeta1_fk_hier_Sc1
# 0.7.75
#       [,1]  [,2]
# [1,] 0.395 0.710
# [2,] 0.365 0.655
# [3,] 0.390 0.680

# 0.6.65.8
#      [,1]  [,2]  [,3]
# [1,] 0.04 0.165 0.935
# [2,] 0.04 0.125 0.920
# [3,] 0.05 0.080 0.935

# 0.85,0.9,0.72
#       [,1] [,2]  [,3]
# [1,] 1.000    1 0.495
# [2,] 1.000    1 0.470
# [3,] 0.995    1 0.540

### CPP

strt <- Sys.time()
Pwr_CPP_store <- NULL
for (i in 1:length(Grid)) {
```

```
  str <- Reject_CPP(Hist = 21, Crnt = 228, Histprob0 = 0.6, Histprob1 =
0.72,
                    Crntprob0 = 0.6, Crntprob1 = Grid[i], n_sims = 200,
                    stancode = "CPP_wout_cov.stan", P = 1)# n_sims = 10 ->
1min
  Pwr_CPP_store <- cbind(Pwr_CPP_store, str)
  cat('grid_num:',i,'in',length(Grid),'\n')
}
tm <- Sys.time()-strt
Pwr_CPP_store
# 0.6
# [1,] 0.055
# 0.65
# [1,] 0.115
# 0.7
# [1,] 0.33
# 0.75
# [1,] 0.645
# 0.8
# [1,] 0.93
# 0.85
# [1,] 0.98
# 0.9.72
# [1,]   1 0.555

### Unnormalized PP
strt <- Sys.time()
Pwr_Unm_PP_store <- NULL
for (i in 1:length(Grid)) {
  str <- Reject_Unnormalized_PP(Hist = 21, Crnt = 228, Histprob0 = 0.6,
Histprob1 = 0.72,
                                Crntprob0 = 0.6, Crntprob1 = Grid[i],
n_sims = 200,
                                stancode = "Unnormal_PP.stan",
                                P = 1, nu = 1, eta = 1)# n_sims = 2 ->
1.5min
  Pwr_Unm_PP_store <- cbind(Pwr_Unm_PP_store, str)
  cat('grid_num:',i,'in',length(Grid),'\n')
}
tm <- Sys.time()-strt
Pwr_Unm_PP_store
# 0.6.65
# [1,] 0.03 0.145
# 0.7.75
# [1,] 0.3 0.72
# 0.8.85
# [1,] 0.905 0.995
# 0.9.72
# [1,]   1 0.5

### Commensurate Prior

set.seed(9999)
n_reps <- 200 # generating 500 times

# For power simulation
Hist_01 = 21
Crnt_01 = 228

Histprob_fk1_0 = 0.6
Histprob_fk1_1 = 0.72
```

```
Crntprob_fk1_0 = 0.6
# Crntprob_fk1_1 = seq(0.45,0.96,by=0.03)
# Crntprob_fk1_1 = seq(0.6, 0.8, by=0.05)
# Crntprob_fk1_1 = seq(0.6, 0.8, by=0.05)
Crntprob_fk1_1 = c(0.85,0.9,0.72)
dput(list(beta0=0, theta0=0),"fake_CP_inits.txt")

pwrbeta1_fk_CP_Sc1 <- rep(0, length(Crntprob_fk1_1))
# pwrbeta1_fk_hier_01 <- rep(0,length(bttau.alpha))

loopstart <- Sys.time()
for (j in 1:length(Crntprob_fk1_1)){
  rejectbeta1 <- 0;
  for (i in 1:n_reps) {
    bt1_new <- NULL
    fake_nnn <- fake(Hist = Hist_01, Crnt = Crnt_01, Histprob0 =
Histprob_fk1_0,
                     Crntprob0 = Crntprob_fk1_0, Histprob1 =
Histprob_fk1_1, Crntprob1 = Crntprob_fk1_1[j])
    # fake_nnn <- fake(Hist = Hist_01, Crnt = Crnt_01, Histprob0 =
Histprob_fk1_0,
    #                  Crntprob0 = Crntprob_fk1_0, Histprob1 =
Histprob_fk1_1, Crntprob1 = Crntprob_fk1_1)
    x0_fake_01 <- fake_nnn$Group[fake_nnn$trialid == "1"]
    y0_fake_01 <- fake_nnn$Timi23[fake_nnn$trialid == "1"]
    x1_fake_01 <- fake_nnn$Group[fake_nnn$trialid == "2"]
    y1_fake_01 <- fake_nnn$Timi23[fake_nnn$trialid == "2"]

    dput(list(x0=as.numeric(x0_fake_01), y0=as.numeric(y0_fake_01),
              x1=as.numeric(x1_fake_01), y1=as.numeric(y1_fake_01),
              n0=Hist_01, n1=Crnt_01, spike= 200, p.choose=0.9,
              l.slab=0.01, u.slab=100), "fake_CP_data.txt")

    modelCheck("CP_BUGS_wout_cov.txt")
    modelData("fake_CP_data.txt")
    modelCompile(numChains=4)
    modelInits("fake_CP_inits.txt")
    modelGenInits()

    modelUpdate(3000)
    samplesSet(c("beta1"))
    modelUpdate(10000)

    # Decision rules
    LB <- samplesStats("beta1")$val2.5pc
    UB <- samplesStats("beta1")$val97.5pc
    if (LB > 0) rejectbeta1 <- rejectbeta1 + 1
    else
      if (UB < 0) rejectbeta1 <- rejectbeta1 + 1
    cat("loop = ",i,"in",n_reps,"\n Grid = ",j,"in",length(Crntprob_fk1_1))
  }
  pwrbeta1_fk_CP_Sc1[j] <- rejectbeta1/n_reps
  cat("\nPr(reject H0: beta_1=0) is: ", pwrbeta1_fk_CP_Sc1[j])
}
tttime <- Sys.time() - loopstart # 2grids * 2loops = 1.3min
pwrbeta1_fk_CP_Sc1
# [1] 0.040 0.165 0.405 0.650 0.940
# [1] 0.995 1.000 0.495

### NPP
strt <- Sys.time()
```

```
Pwr_NPP_store_Sc1 <- NULL
for (i in 1:length(Grid)) {
  str <- Reject_NPP(Hist = 21, Crnt = 228, Histprob0 = 0.6, Histprob1 =
0.72,
                    Crntprob0 = 0.6, Crntprob1 = Grid[i], n_sims = 200,
                    prr_stancode = "logistic_regression_prior.stan",
                    postrr_stancode = "NPP_approximate.stan",
                    P = 1, nu = 1, eta = 1)# n_sims = 1 -> 3min
  Pwr_NPP_store_Sc1 <- cbind(Pwr_NPP_store_Sc1, str)
  cat('grid_num:',i,'\n')
}
tm <- Sys.time()-strt
Pwr_NPP_store_Sc1



######### Scenario 3: Imb 0.8 0.92 0.6 Grids
########################################

# GLM
Grid <- c(seq(0.6, 0.9, by=0.05), 0.72)
Rslt_pwr_store_glm_Sc3 <- NULL
for (i in 1:length(Grid)) {
  str <- fake_rej_glm1(Hist = 21, Crnt = 228, Histprob0 = 0.8, Crntprob0 =
0.6, Histprob1 = 0.92, Crntprob1 = Grid[i], n_sims = 1000)
  Rslt_pwr_store_glm_Sc3 <- cbind(Rslt_pwr_store_glm_Sc3, str)
}
Rslt_pwr_store_glm_Sc3
# [1,] 0.043 0.149 0.398 0.71 0.949 0.995   1 0.52


Grid <- c(seq(0.6,0.9,by=0.05),0.72)
# Two meta
Rslt_pwr_store_twometa_Sc3 <- NULL
for (i in 1:length(Grid)) {
  str <- fake_rej_twometa(Hist = 228, Crnt = 228, Histprob0 = 0.8,
Crntprob0 = 0.6, Histprob1 = 0.92, Crntprob1 = Grid[i], n_sims = 1000)
  Rslt_pwr_store_twometa_Sc3 <- cbind(Rslt_pwr_store_twometa_Sc3, str)
}
Rslt_pwr_store_twometa_Sc3
# [1,] 0.063 0.207 0.491 0.782 0.904 0.933 0.85 0.64

# PQL
Grid <- 0.72
Rslt_pwr_store_PQL_Sc3 <- NULL
for(i in 1:length(Grid)){
  str <- fake_rej_PQL(Hist = 21, Crnt = 228, Histprob0 = 0.8, Crntprob0 =
0.6, Histprob1 = 0.92, Crntprob1 = Grid[i], n_sims = 1000)
  Rslt_pwr_store_PQL_Sc3 <- cbind(Rslt_pwr_store_PQL_Sc3, str)
}
Rslt_pwr_store_PQL_Sc3
# [1,] 0.053 0.144 0.419 0.719 0.929 0.975 NA 0.532

# Unnormalized_PP
strt <- Sys.time()
Pwr_Unm_PP_store <- NULL
for (i in 1:length(Grid)) {
  str <- Reject_Unnormalized_PP(Hist = 21, Crnt = 228, Histprob0 = 0.8,
Histprob1 = 0.92,
                                Crntprob0 = 0.6, Crntprob1 = Grid[i],
n_sims = 200,
```

```
                                stancode = "Unnormal_PP.stan",
                                P = 1, nu = 1, eta = 1)# n_sims = 2 ->
1.5min
  Pwr_Unm_PP_store <- cbind(Pwr_Unm_PP_store, str)
  cat('grid_num:',i,'in',length(Grid),'\n')
}
tm <- Sys.time()-strt
Pwr_Unm_PP_store
# 0.6.65
# [1,] 0.05 0.155
# 0.7.75
# [1,] 0.39 0.755
# 0.8.85
# [1,] 0.91 0.98
# 0.9.72
# [1,]   1 0.515

### CP Sc3

set.seed(9999)
n_reps <- 200 # generating 500 times

# For power simulation
Hist_01 = 21
Crnt_01 = 228

Histprob_fk1_0 = 0.8
Histprob_fk1_1 = 0.92
Crntprob_fk1_0 = 0.6
# Crntprob_fk1_1 = seq(0.45,0.96,by=0.03)
# Crntprob_fk1_1 = seq(0.6, 0.8, by=0.05)
# Crntprob_fk1_1 = seq(0.6, 0.8, by=0.05)
Crntprob_fk1_1 = c(0.85,0.9,0.72)
dput(list(beta0=0, theta0=0),"fake_CP_inits.txt")

pwrbeta1_fk_CP_Sc3 <- rep(0, length(Crntprob_fk1_1))
# pwrbeta1_fk_hier_01 <- rep(0,length(bttau.alpha))

loopstart <- Sys.time()
for (j in 1:length(Crntprob_fk1_1)){
  rejectbeta1 <- 0;
  for (i in 1:n_reps) {
    bt1_new <- NULL
    fake_nnn <- fake(Hist = Hist_01, Crnt = Crnt_01, Histprob0 =
Histprob_fk1_0,
                     Crntprob0 = Crntprob_fk1_0, Histprob1 =
Histprob_fk1_1, Crntprob1 = Crntprob_fk1_1[j])
    # fake_nnn <- fake(Hist = Hist_01, Crnt = Crnt_01, Histprob0 =
Histprob_fk1_0,
    #                  Crntprob0 = Crntprob_fk1_0, Histprob1 =
Histprob_fk1_1, Crntprob1 = Crntprob_fk1_1)
    x0_fake_01 <- fake_nnn$Group[fake_nnn$trialid == "1"]
    y0_fake_01 <- fake_nnn$Timi23[fake_nnn$trialid == "1"]
    x1_fake_01 <- fake_nnn$Group[fake_nnn$trialid == "2"]
    y1_fake_01 <- fake_nnn$Timi23[fake_nnn$trialid == "2"]

    dput(list(x0=as.numeric(x0_fake_01), y0=as.numeric(y0_fake_01),
              x1=as.numeric(x1_fake_01), y1=as.numeric(y1_fake_01),
              n0=Hist_01, n1=Crnt_01, spike= 200, p.choose=0.9,
              l.slab=0.01, u.slab=100), "fake_CP_data.txt")
```

```
    modelCheck("CP_BUGS_wout_cov.txt")
    modelData("fake_CP_data.txt")
    modelCompile(numChains=4)
    modelInits("fake_CP_inits.txt")
    modelGenInits()

    modelUpdate(3000)
    samplesSet(c("beta1"))
    modelUpdate(10000)

    # Decision rules
    LB <- samplesStats("beta1")$val2.5pc
    UB <- samplesStats("beta1")$val97.5pc
    if (LB > 0) rejectbeta1 <- rejectbeta1 + 1
    else
      if (UB < 0) rejectbeta1 <- rejectbeta1 + 1
    cat("loop = ",i,"in",n_reps,"\n Grid = ",j,"in",length(Crntprob_fk1_1))
  }
  pwrbeta1_fk_CP_Sc3[j] <- rejectbeta1/n_reps
  cat("\nPr(reject H0: beta_1=0) is: ", pwrbeta1_fk_CP_Sc3[j])
}
tttime <- Sys.time() - loopstart # 2grids * 2loops = 1.3min
pwrbeta1_fk_CP_Sc3
# [1] 0.04 0.18 0.43 0.69 0.94
# [1] 1.00 1.00 0.55


### CPP Sc3

# 0.6.65
# [1,] 0.04 0.135
# 0.7.75
# [1,] 0.37 0.675
# 0.8.85
# [1,] 0.865 0.985
# 0.9.72
# [1,]   1 0.57


### Unnormal_PP
strt <- Sys.time()
Pwr_Unm_PP_store_Sc3 <- NULL
for (i in 1:length(Grid)) {
  str <- Reject_Unnormalized_PP(Hist = 21, Crnt = 228, Histprob0 = 0.8,
Histprob1 = 0.92,
                                Crntprob0 = 0.6, Crntprob1 = Grid[i],
n_sims = 200,
                                stancode = "Unnormal_PP.stan",
                                P = 1, nu = 1, eta = 1)# n_sims = 2 ->
1.5min
  Pwr_Unm_PP_store_Sc3 <- cbind(Pwr_Unm_PP_store_Sc3, str)
}
tm <- Sys.time()-strt
Pwr_Unm_PP_store_Sc3

### Std BHM
# 0.6.65.7
#       [,1]  [,2]  [,3]
# [1,] 0.040 0.170 0.375
# [2,] 0.035 0.125 0.325
# [3,] 0.055 0.000 0.515
```

```
# 0.75.8.85
#       [,1]  [,2]  [,3]
# [1,] 0.680 0.910 0.985
# [2,] 0.660 0.935 0.995
# [3,] 0.68 0.935 0.995

# 0.9.72
#       [,1]  [,2]
# [1,] 1.000 0.520
# [2,] 0.995 0.485
# [3,] 1.000 0.485

######### Meta results ##########################

# Construct the dataset(Mean + 95(B)CI)
Mean_overall <- NULL
CI_overall <- NULL

# Two-stage results
coef(fit2_2_14) # 0.2049694
confint(fit2_2_14)
#                 2.5 %    97.5 %
# (Intercept) -0.311923 0.7218619

coef(fit2_2_24) # 0.5574177
confint(fit2_2_24)
#                    2.5 %   97.5 %
#   (Intercept) 0.02273217 1.092103

# One-stage results
fixef(IPD14)["Group"] # 0.2049719
confint(IPD14)
#                  2.5 %    97.5 %
# .sig01       0.0000000 0.8297065
# (Intercept) -0.6164542 3.3232640
# trialid     -1.4805968 0.5246239
# Group       -0.4018611 0.8142913


fixef(IPD24)["Group"] # 0.5574595
confint(IPD24)
#                   2.5 %    97.5 %
# .sig01       0.00000000 0.9065761
# (Intercept) -0.53559165 3.6661617
# trialid     -1.65686785 0.4831038
# Group       -0.06600734 1.2414066

# GLM for direct pooling
coef(fit_dir_ab14)["Group"] # 0.2049719
confint(fit_dir_ab14)
#                  2.5 %    97.5 %
# (Intercept) -0.6163845 3.3235454
# trialid     -1.4807296 0.5245770
# Group       -0.3114115 0.7240712


coef(fit_dir_ab24)["Group"] # 0.5574595
confint(fit_dir_ab24)
#                   2.5 %    97.5 %
# (Intercept) -0.53572677 3.6664531
```

```
# trialid     -1.65702671 0.4830719
# Group        0.02609251 1.0976075

# GLM for original datasets
coef(fit_vir_14)["Group"] # 0.2107137
confint(fit_vir_14)
#                  2.5 %    97.5 %
# (Intercept)  0.02011276 0.7706708
# Group       -0.32596817 0.7504989

coef(fit_vir_24)["Group"] # 0.550117
confint(fit_vir_24)
#                     2.5 %    97.5 %
# (Intercept)  0.0201127591 0.7706708
# Group       -0.0004550502 1.1099176

# SHM direct pooling

# For 14
# stats for beta1:
#          mean     sd  MC_error val2.5pc median val97.5pc start  sample
tau_alpha[j] tau_beta[j] tau_alpha0[k] tau_beta0[k] beta1id
# beta1  0.2094 0.2633 0.0001467  -0.3060 0.2091    0.7267 50001 5000000
2           2             2            2       1
# beta11 0.2093 0.2635 0.0001415  -0.3064 0.2089    0.7270 50001 5000000
6           6             2            2       2
# beta12 0.2093 0.2634 0.0001400  -0.3065 0.2090    0.7264 50001 5000000
16          16             2            2       3

# For 24
# stats for beta1:
#         mean     sd  MC_error val2.5pc median val97.5pc start  sample
tau_alpha[j] tau_beta[j] tau_alpha0[k] tau_beta0[k] beta1id
# beta1 0.5619 0.2725 0.0001575  0.03042 0.5609     1.099 50001 5000000
2           2             2            2       1


# Hierarchical Modeling

# For 14
# stats for beta1:
#          mean     sd  MC_error val2.5pc median val97.5pc start  sample
bttau.alpha[j] bttau.beta[j] thttau.alpha[k] thttau.beta[k] beta1id
# beta1  0.2056 0.2699 0.0001572  -0.3223 0.2052    0.7364 50001 5000000
2            2               2              2       1
# beta11 0.2049 0.2705 0.0001508  -0.3238 0.2046    0.7368 50001 5000000
6            6               2              2       2
# beta12 0.2045 0.2707 0.0001478  -0.3252 0.2041    0.7367 50001 5000000
16           16              2              2       3

# For 24
# stats for beta1:
#          mean     sd  MC_error val2.5pc median val97.5pc start  sample
bttau.alpha[j] bttau.beta[j] thttau.alpha[k] thttau.beta[k] beta1id
# beta1  0.5531 0.2785 0.0001647 0.010120 0.5519     1.103 50001 5000000
2            2               2              2       1
# beta11 0.5516 0.2791 0.0001609 0.007483 0.5506     1.102 50001 5000000
6            6               2              2       2
# beta12 0.5515 0.2794 0.0001579 0.007417 0.5502     1.103 50001 5000000
16           16              2              2       3
```

```
# Power Prior
# For 14

# Approximate NPP
# [[4]] Ks=10000
#                mean      se_mean          sd         2.5%          25%
50%         75%       97.5%    n_eff     Rhat
# alpha     0.4201615 1.390122e-03 0.177486702   0.07535776   0.29964368
0.4181441   0.5399722   0.7700407 16301.44 1.000094
# beta[1]   0.2103654 1.981393e-03 0.253886658  -0.28697128   0.03909049
0.2103112   0.3805863   0.7096814 16418.69 1.000188
# a_0       0.9901102 6.377946e-05 0.009973895   0.96283003   0.98634989
0.9932248   0.9971710   0.9997009 24455.01 1.000005
# lp__     -26.8332911 1.120082e-02 1.244304652 -29.99084489 -27.42653038
-26.5235550 -25.9116403 -25.3840247 12341.09 1.000173


# Unnormalised PP
#                  mean      se_mean          sd          2.5%           25%
50%          75%        97.5%    n_eff     Rhat
# alpha      0.38887061 0.0013950393 0.18501318  3.010371e-02    0.26512917
0.38717810    0.5128090    0.7553664 17588.65 1.000042
# beta[1]    0.21312628 0.0020209709 0.26427692 -3.029545e-01    0.03562587
0.21155746    0.3899991    0.7358875 17100.09 1.000034
# a_0        0.07677404 0.0004869426 0.07689569  2.024784e-03    0.02185407
0.05308495    0.1066334    0.2822735 24937.24 1.000059
# lp__     -157.48514240 0.0115693129 1.29826065 -1.608298e+02 -158.07974426
-157.14048860 -156.5312202 -156.0057746 12592.40 1.000069

# For 24

# Approximate NPP
# [[4]] Ks=10000
#                mean      se_mean          sd         2.5%          25%
50%         75%       97.5%    n_eff     Rhat
# alpha     0.4321511 0.0013502065 0.177041611   0.08633824   0.3126189
0.4306848   0.5520955   0.7819086 17192.95 1.000152
# beta[1]   0.5405757 0.0020077549 0.261021060   0.03115054   0.3652795
0.5407330   0.7161924   1.0572998 16901.67 1.000088
# a_0       0.9901290 0.0000647023 0.009889735   0.96362176   0.9864106
0.9931633   0.9971523   0.9996907 23363.07 1.000130
# lp__     -20.0800232 0.0110467845 1.221752116 -23.18251591 -20.6706524
-19.7783263 -19.1733858 -18.6484407 12231.91 1.000264

# Unnormalized PP
#                  mean      se_mean          sd          2.5%           25%
50%          75%        97.5%    n_eff      Rhat
# alpha      0.39973177 0.0014142085 0.18432693  3.910070e-02    0.27619230
0.39931059    0.5223905    0.7657500 16988.33 0.9999899
# beta[1]    0.53033348 0.0020809688 0.26938825  4.851858e-03    0.34905383
0.52926926    0.7110225    1.0643246 16758.15 1.0001853
# a_0        0.08386243 0.0005217527 0.08286563  2.216849e-03    0.02474549
0.05817665    0.1161353    0.3074494 25224.32 0.9999673
# lp__     -151.74777607 0.0114817641 1.28026241 -1.550596e+02 -152.32837994
-151.41618727 -150.8082999 -150.2824408 12433.14 1.0000223

# Commensurate PP w/out normalization

# For 14
```

```
#                  mean      se_mean         sd          2.5%           25%
50%           75%        97.5%    n_eff      Rhat
# alpha0      0.18093429 0.0048998797 0.9135335 -1.634848e+00   -0.42869614
0.19293044    0.80104612    1.9542977 34759.87 0.9999516
# alpha1      0.38857968 0.0011515902 0.1855128  2.916186e-02    0.26275058
0.38647756    0.51166984    0.7555672 25950.88 0.9999825
# beta0[1]    0.09632026 0.0051018918 0.9624809 -1.797980e+00   -0.54360224
0.09687862    0.75115502    1.9816751 35589.49 0.9999117
# beta1[1]    0.20905019 0.0016417314 0.2642349 -3.080560e-01    0.03099103
0.20998184    0.38576958    0.7311837 25904.56 0.9999629
# a_0         0.06443360 0.0003544153 0.0658810  1.743154e-03    0.01837094
0.04395477    0.08852673    0.2448044 34553.78 0.9999698
# lp__      -160.39732992 0.0137389391 1.6242298 -1.644627e+02 -161.24509279
-160.06507421 -159.20294078 -158.2455928 13976.18 1.0001751

# For 24

#                  mean      se_mean         sd          2.5%
25%           50%           75%        97.5%    n_eff      Rhat
# alpha0      0.21697609 0.0049499971 0.91710653 -1.613408e+00
-0.38868125    0.22686237    0.83184772    1.9944183 34326.51 0.9999937
# alpha1      0.39968644 0.0011239191 0.18552505  3.958867e-02
0.27392684    0.39735012    0.52326750    0.7679532 27248.02 0.9999233
# beta0[1]    0.14621650 0.0051082987 0.95040053 -1.722637e+00
-0.49529973    0.14944255    0.78969219    1.9965437 34614.72 0.9999483
# beta1[1]    0.52621817 0.0016390350 0.27132098 -7.828390e-03
0.34371440    0.52518067    0.70903875    1.0587529 27402.51 0.9999872
# a_0         0.06834159 0.0003829109 0.07159254  1.710514e-03
0.01866525    0.04548824    0.09366823    0.2637218 34957.48 0.9999130
# lp__      -154.72702248 0.0137363814 1.62508326 -1.587247e+02
-155.56900755 -154.40213979 -153.53842455 -152.5657009 13996.08 1.0004754

# Commensurate PP with Normalization (Real cpp)

# For 14
# [[4]] Ks=10000
#                 mean      se_mean          sd        2.5%          25%
50%          75%       97.5%    n_eff      Rhat
# alpha0     0.6845354 3.171796e-03 0.511633253  -0.2975110   0.33802443
0.6764988   1.0268486   1.7094793 26019.99 0.9999170
# alpha1     0.3890381 1.204153e-03 0.184514323   0.0287385   0.26470295
0.3876648   0.5122426   0.7557588 23479.95 1.0001083
# beta0[1]   0.2554602 4.078858e-03 0.666722302  -1.0377987  -0.19798215
0.2539251   0.7009601   1.5692606 26718.54 0.9999668
# beta1[1]   0.2086555 1.740838e-03 0.261947110  -0.3100282   0.03489697
0.2074753   0.3855007   0.7209385 22641.77 1.0001374
# a_0        0.9898818 5.281426e-05 0.009947829   0.9631778   0.98599231
0.9929018   0.9969737   0.9996714 35477.60 1.0000169
# lp__      -29.5679668 1.324592e-02 1.589534278 -33.4471523 -30.40493129
-29.2510716 -28.3858016 -27.4397469 14400.44 1.0001278

# For 24
# [[4]] Ks=10000
#                 mean      se_mean          sd          2.5%          25%
50%          75%       97.5%    n_eff      Rhat
# alpha0     0.7738944 3.155370e-03 0.523594046  -0.228442093   0.41842581
0.7665374   1.1211672   1.8256944 27535.24 1.0001353
# alpha1     0.3976925 1.225334e-03 0.185340498   0.037338881   0.27357318
0.3967160   0.5222272   0.7625334 22878.72 0.9999680
# beta0[1]   0.5353916 4.195883e-03 0.690582755  -0.801232484   0.06751425
0.5272977   0.9971265   1.8980618 27088.48 1.0000208
```

```
# beta1[1]   0.5306469 1.771682e-03 0.272051572   0.002178506   0.34433360
0.5289936   0.7172397   1.0617387 23579.31 0.9999629
# a_0        0.9900483 5.307472e-05 0.009806462   0.964089129   0.98615168
0.9930327   0.9970625   0.9997115 34138.88 1.0000048
# lp__      -22.9296261 1.338257e-02 1.608836606 -26.883982709 -23.77693356
-22.6027568 -21.7437055 -20.7833124 14452.57 1.0002097


######### Forest plot ##########################################

ggplot(
  beta_by_subj,
  aes(x = Estimate, xmin = Q2.5, xmax = Q97.5, y = subject)) +
  geom_point() +
  geom_errorbarh() +
  geom_vline(xintercept = grand_av_beta$mean) +
  geom_vline(xintercept = grand_av_beta$lq, linetype = "dashed") +
  geom_vline(xintercept = grand_av_beta$hq, linetype = "dashed") +
  xlab("By-subject effect of cloze probability in microvolts")

########## NPP function in one-shot
########################################
fake <- function(Hist, Crnt, Histprob0, Histprob1, Crntprob0, Crntprob1){
  trialid <- c(rep(1, Hist), rep(2, Crnt)) # the same as the 2-4 of a+b

  Hist_Cntrl <- round(Hist/2)
  Hist_Trt <- ceiling(Hist/2)
  Crnt_Cntrl <- round(Crnt/2)
  Crnt_Trt <- ceiling(Crnt/2)

  Tm23_Hist_Cntrl <- c(rbinom(Hist_Cntrl, 1, Histprob0))
  Tm23_Hist_Trt <- c(rbinom(Hist_Trt, 1, Histprob1))
  Tm23_Crnt_Cntrl <- c(rbinom(Crnt_Cntrl, 1, Crntprob0))
  Tm23_Crnt_Trt <- c(rbinom(Crnt_Trt, 1, Crntprob1))
  # Grp <- c(Group_Hist, Group_Crnt)
  Group <- c(rep(0, Hist_Cntrl), rep(1, Hist_Trt), rep(0, Crnt_Cntrl),
rep(1, Crnt_Trt))
  Timi23 <- c(Tm23_Hist_Cntrl, Tm23_Hist_Trt, Tm23_Crnt_Cntrl,
Tm23_Crnt_Trt)

  return(data.frame(Timi23, Group, trialid))
}


get_posterior_K <- function(K, data_postrr, constt_data, model,
approx_model, iter = 13000, warmup = 3000){
  pred_a0s <- seq(0, max(constt_data$a0), length.out = K) # splitting
points
  a0_grid <- data.frame(a0 = pred_a0s,
                        lc_pred = predict(approx_model, newdata =
data.frame(a0 = pred_a0s)))
  data_postrr$pred_grid_x <- a0_grid$a0
  data_postrr$pred_grid_y <- a0_grid$lc_pred
  approx.norm.posterior.lgr <- sampling(model,
                                        data = data_postrr, refresh = 0,
                                        iter = iter, warmup = warmup)
  sumry_stat <- summary(approx.norm.posterior.lgr)
  cat('posterior generated\n')
  return(sumry_stat)
}
```

```
Approximation_Normalized_PP <- function(J=20, maxA=1, epsilon=0.05,
prior_stancode,
                                        postrr_stancode, data_a0,
data_postrr, Ks=5000){

  logistic_prior <- stan_model(prior_stancode)
  adaptive.ca0.estimates <- build_grid(compiled.model.prior =
logistic_prior, eps = epsilon,
                                       M = maxA, J = J, v1 = 10, v2 = 10,
                                       stan.list = data_a0, pars =
c("alpha", "beta"))
  constant_data <- adaptive.ca0.estimates$result

  fit.gam <- mgcv::gam(lc_a0 ~ s(a0, k = J + 1), data = constant_data) #
Spline smoothing function Generalized Additive Model.
  cat('approximate constant data generated\n')
  approx.normalised.model <- stan_model(postrr_stancode)

  all.approximates <- get_posterior_K(K=Ks, model=approx.normalised.model,
                                      constt_data=constant_data,
data_postrr = data_postrr, approx_model = fit.gam)
  return(all.approximates)
}


Reject_NPP <- function(Hist, Crnt, Histprob0, Histprob1, Crntprob0,
Crntprob1, prr_stancode, postrr_stancode, P, nu, eta, n_sims = 500){
  rej <- 0
  for (i in 1:n_sims) {
    fake_PP <- fake(Hist, Crnt, Histprob0, Histprob1, Crntprob0, Crntprob1)
    N_Hist_PP <- Hist
    X_Hist_PP <- matrix(NA, nrow = N_Hist_PP, ncol = P)
    X_Hist_PP <- fake_PP$Group[fake_PP$trialid == "1"]
    Y_Hist_PP <- fake_PP$Timi23[fake_PP$trialid == "1"]
    X_Hist_PP <- as.matrix(X_Hist_PP)

    ### will draw from the same process for simplicity
    N_Cur_PP <- Crnt
    X_Cur_PP <- matrix(NA, nrow = N_Cur_PP, ncol = P)
    X_Cur_PP <- fake_PP$Group[fake_PP$trialid == "2"]
    Y_Cur_PP <- fake_PP$Timi23[fake_PP$trialid == "2"]
    X_Cur_PP <- as.matrix(X_Cur_PP)

    lgr_fk_data <- list(
      N0 = N_Hist_PP,
      X0 = X_Hist_PP,
      P = P,
      y0 = Y_Hist_PP,
      eta = eta,
      nu = nu,
      X = X_Cur_PP,
      N = N_Cur_PP,
      y = Y_Cur_PP
    )
    lgr_fk_fora0_data <- list(
      N0 = N_Hist_PP,
      X0 = X_Hist_PP,
      P = P,
      y0 = Y_Hist_PP,
      a_0 = NULL
    )
```

```
    cat('data generated\n')
    Approximate_NPP_rslt_fk <- Approximation_Normalized_PP(prior_stancode =
prr_stancode,
                                                           postrr_stancode
= postrr_stancode,
                                                           data_postrr =
lgr_fk_data,
                                                           data_a0 =
lgr_fk_fora0_data)
    LB <- Approximate_NPP_rslt_fk$summary[2,4]
    UB <- Approximate_NPP_rslt_fk$summary[2,8]
    if(LB > 0) rej = rej + 1
    else
      if(UB < 0) rej = rej + 1

    cat('loops = ', i, 'in', n_sims, '\n', 'loop_time =', Sys.time()-strt,
'\n')
  }
  rej_rate <- rej/n_sims
  return(rej_rate)
}

# apply parallel frameworks
# detectCores()
plan(multisession)

Grid <- c(0.6, 0.65, 0.7)

# storing the result
Para_rslt_NPP_Sc1 <- data.frame(Grids = numeric(), Power = numeric(),
stringsAsFactors = FALSE)

# the parallel function
compute_results <- function(Grid) {

  Pwr_NPP_store <- NULL
  str <- Reject_NPP(Hist = 21, Crnt = 228, Histprob0 = 0.7, Histprob1 =
0.82,
                    Crntprob0 = 0.6, Crntprob1 = Grid, n_sims = 10,
                    prr_stancode = "logistic_regression_prior.stan",
                    postrr_stancode = "NPP_approximate.stan",
                    P = 1, nu = 1, eta = 1)
  Pwr_NPP_store <- cbind(Pwr_NPP_store, str)
  cat('grid finished! \n')
  flush.console()

  return(list(Power = Pwr_NPP_store))
}

# parallel computing
strt <- Sys.time()
results <- future_lapply(Grid, compute_results)
tm <- Sys.time() - strt

# restore the results
for (i in seq_along(results)) {
  Para_rslt_NPP_Sc1[i, "Grids"] <- Grid[i]
  Para_rslt_NPP_Sc1[i, "Power"] <- results[[i]]$Power
}

print(Para_rslt_NPP_Sc1)
```

```
# Results
# Sc1_g1_1
# Grids Power
# 1  0.60 0.125
# 2  0.65 0.100
# 3  0.70 0.350
# 4  0.75 0.700
# Time difference of 8.565517 hours

# Sc1_g2_1
# Grids Power
# 1  0.80 0.975
# 2  0.85 1.000
# 3  0.90 0.975
# 4  0.72 0.625
# Time difference of 8.322705 hours

# Sc3_g1_1
# Grids Power
# 1  0.60 0.050
# 2  0.65 0.225
# 3  0.70 0.325
# 4  0.75 0.675
# Time difference of 7.822768 hours

# Sc3_g2_1
# Grids Power
# 1  0.80 0.800
# 2  0.85 1.000
# 3  0.90 1.000
# 4  0.72 0.475
# Time difference of 7.585191 hours

# Sc1_g1_2
# Grids Power
# 1  0.60 0.075
# 2  0.65 0.100
# 3  0.70 0.325
# 4  0.75 0.775
# Time difference of 7.882012 hours

# Sc1_g2_2
# Grids Power
# 1  0.80 0.875
# 2  0.85 1.000
# 3  0.90 1.000
# 4  0.72 0.475
# Time difference of 7.830264 hours

# Sc3_g1_2
# Grids Power
# 1  0.60 0.000
# 2  0.65 0.075
# 3  0.70 0.450
# 4  0.75 0.650
# Time difference of 8.142253 hours

# Sc3_g2_2
# Grids Power
```

```
# 1  0.80 0.950
# 2  0.85 0.975
# 3  0.90 1.000
# 4  0.72 0.425
# Time difference of 7.962359 hours

# Sc1_g1_3
# Grids Power
# 1  0.60 0.075
# 2  0.65 0.225
# 3  0.70 0.475
# 4  0.75 0.625
# Time difference of 8.237582 hours

# Sc1_g2_3
# Grids Power
# 1  0.80 0.875
# 2  0.85 0.975
# 3  0.90 1.000
# 4  0.72 0.525
# Time difference of 9.099135 hours

# Sc3_g1_3
# Grids Power
# 1  0.60 0.000
# 2  0.65 0.225
# 3  0.70 0.500
# 4  0.75 0.675
# Time difference of 8.331921 hours

# Sc3_g2_3
# Grids Power
# 1  0.80  0.95
# 2  0.85  1.00
# 3  0.90  1.00
# 4  0.72  0.55
# Time difference of 7.388308 hours

# Sc1_g1_4
# Grids Power
# 1  0.60 0.025
# 2  0.65 0.100
# 3  0.70 0.350
# 4  0.75 0.825
# Time difference of 8.302962 hours

# Sc1_g2_4
# Grids Power
# 1  0.80 0.900
# 2  0.85 1.000
# 3  0.90 1.000
# 4  0.72 0.475
# Time difference of 7.614323 hours

# Sc3_g1_4
# Grids Power
# 1  0.60 0.000
# 2  0.65 0.250
# 3  0.70 0.350
# 4  0.75 0.675
# Time difference of 7.298785 hours
```

```
# Sc3_g2_4
# Grids Power
# 1  0.80  0.95
# 2  0.85  1.00
# 3  0.90  1.00
# 4  0.72  0.55
# Time difference of 7.400093 hours

# Sc1_g1_5
# Grids Power
# 1  0.60 0.075
# 2  0.65 0.150
# 3  0.70 0.600
# 4  0.75 0.750
# Time difference of 9.504329 hours

# Sc1_g2_5
# Grids Power
# 1  0.80   0.9
# 2  0.85   1.0
# 3  0.90   1.0
# 4  0.72   0.5
# Time difference of 8.14723 hours

# Sc3_g1_5
# Grids Power
# 1  0.60 0.100
# 2  0.65 0.075
# 3  0.70 0.325
# 4  0.75 0.825
# Time difference of 7.343722 hours

# Sc3_g2_5
# Grids Power
# 1  0.80 0.950
# 2  0.85 1.000
# 3  0.90 1.000
# 4  0.72 0.575
# Time difference of 7.366919 hours

### Sc1 results
# 0.6: 0.075
# 0.65: 0.135
# 0.7: 0.42
# 0.75: 0.735
# 0.8: 0.905
# 0.85: 0.995
# 0.9: 0.995
# 0.72: 0.52

### Sc3 results:
# 0.6: 0.03
# 0.65: 0.17
# 0.7: 0.39
# 0.75: 0.7
# 0.8: 0.92
# 0.85: 0.995
# 0.9: 1.000
# 0.72: 0.515
```

######### Ending ##########################