Supporting Information S5

R code for simulating data according to 'check the positives' design and fitting adjustment model for $sd = 0.01$ using a correctly centred prior for B_{sp}.

```
library(foreach)
library(rjags)
require(parallel)
require(doParallel)

# Define parameter values
n_vals <- c(100)
pi_vals <- c(0.55)
A_se_vals <- c(0.75)
A_sp_vals <- c(0.8)
B_se_vals <- c(0.9)

# Number of simulations for each parameter combination
n_sim <- 400

# Grid of all parameter combinations
param_grid <- expand.grid(
  n = n_vals,
  pi = pi_vals,
  A_se = A_se_vals,
  A_sp = A_sp_vals,
  B_se = B_se_vals
)

# Function to generate data from multinomial distribution
data_gen <- function(pi, A_se, A_sp, B_se, n_samples, seed) {
  set.seed(seed)  # Set seed for reproducibility within each worker

  p1 <- pi*(A_se * B_se)                          # TP
  p2 <- pi*((1-A_se) * B_se)                       # FN
  p3 <- (1-pi)*(1-A_sp) + pi*A_se*(1-B_se)         # FP
  p4 <- (1-pi)*A_sp + pi*(1-A_se)*(1-B_se)         # TN

  p <- c(p1, p2, p3, p4)  # Combine probabilities

  if (abs(sum(p) - 1) > 1e-6) {
    stop("Probabilities do not sum to 1. Check your inputs.")
  }
```

```
  d <- rmultinom(1, n_samples, p)  # Generate data
  return(d)
}

# Set number of chains, before and after burn in iterations
n.chains <- 3
burnin.it <- 30000
sample.it <- 1500000

# Define the JAGS model function
JAGS_model <- function(row, sd) {
  # Correctly centered prior
  mean <- row$B_se
  v <- sd^2
  b <- (mean * (1 - mean)^2) / v + mean - 1
  a <- (mean * b) / (1 - mean)

  datalist <- list(M = c(row$TP, row$FN, row$FP, row$TN),
# TP, FN, FP, TN
                   a = a,
                   b = b,
                   N = row$n)

  model_string <- "model{

  M[1:4] ~ dmulti(p[1:4], N) # multinomial likelihood for the 2x2 data

  # Model for p[1:4], assuming conditional independence:

  p[1] <- pi*(A_se * B_se) #TP

  p[2] <- pi*((1-A_se) * B_se) #FN

  p[3] <- (1-pi)*(1-A_sp) + pi*A_se*(1-B_se) #FP

  p[4] <- (1-pi)*A_sp + pi*(1-A_se)*(1-B_se) #TN

  # Informative prior for specificity of the reference test:
  B_se ~ dbeta(a, b)I(0,0.9999) # Beta prior distribution

  # Vague priors for the other parameters:
  pi ~ dbeta(1,1) # prevalence
  A_se ~ dbeta(1, 1) # sensitivity of the index test
  A_sp ~ dbeta(1, 1) # specificity of the index test
```

```
}"

  model <- jags.model(file = textConnection(model_string),
                      data = datalist, n.chains = n.chains)
  update(model, n.iter = burnin.it)

  posterior_sample <- coda.samples(model,
                                   variable.names = c("A_se", "A_sp", "B_se", "p
                                   n.iter = sample.it)

  summary_stats <- summary(posterior_sample)

  gel_diag <- gelman.diag(posterior_sample)$psrf[1:4]
  neff <- effectiveSize(posterior_sample)


  # Return relevant statistics as a named list
  return(list(
    adjusted_sp = summary_stats$quantiles["A_sp", "50%"],
    A_sp_SD = summary_stats$statistics["A_sp", "SD"],
    lower_sp = summary_stats$quantiles["A_sp", "2.5%"],
    higher_sp = summary_stats$quantiles["A_sp", "97.5%"],
    gr = all(gel_diag <= 1.1),
    neff = all(neff >= 2500)
  ))
}


# Set up parallel backend
cl <- makeCluster(12)
registerDoParallel(cl)

base_seed <- 12345

# Parallel execution with multiple simulations per parameter combination
results <- foreach(i = 1:nrow(param_grid), .packages = c("rjags"), .combine = rb
  row <- param_grid[i, ]
  sim_results <- data.frame()  # Initialize an empty data frame for this paramet

  for (sim in 1:n_sim) {
    # Generate data
    counts <- data_gen(row$pi, row$A_se, row$A_sp, row$B_se, row$n, base_seed +
    data <- data.frame(
      n = row$n,
      pi = row$pi,
      A_se = row$A_se,
```

```
      A_sp = row$A_sp,
      B_se = row$B_se,
      TP = counts[1, 1],
      FN = counts[2, 1],
      FP = counts[3, 1],
      TN = counts[4, 1]
    )

    # Run JAGS model
    res <- JAGS_model(data, sd = 0.01)

    # Combine results for this simulation
    sim_result <- data.frame(
      n = row$n,
      pi = row$pi,
      A_se = row$A_se,
      A_sp = row$A_sp,
      B_se = row$B_se,
      TP = data$TP,
      FN = data$FN,
      FP = data$FP,
      TN = data$TN,
      adjusted_sp = res$adjusted_sp,
      A_sp_SD = res$A_sp_SD,
      lower_sp = res$lower_sp,
      higher_sp = res$higher_sp,
      gr = res$gr,
      neff = res$neff,
      sd = 0.01,
      mean_B_se = row$B_se)

    sim_results <- rbind(sim_results, sim_result)  # Append results
  }

  return(sim_results)
}

# Stop cluster
stopCluster(cl)
```