Supplements

Contents

Derivation of the dependence of relative DIF in item intercepts on identification restrictions on item slopes

In the following, we will show that relative DIF in item intercepts depends on the identification restriction on item slopes.

Relative β-DIF R_{ij}^{β} for item i and j for two groups is defined as:

$$R_{ij}^{\beta} = (\beta_{i1} - \beta_{j1}) - (\beta_{i2} - \beta_{j2}). \tag{1}$$

Within structural equation modeling, item intercepts β_{ig} are identified based on the underlying latent response variable y_{ig}^{*} as follows:

$$\beta_{ig} = E(y_{ig}^{*}) - \alpha_{ig}E(\theta_g) \tag{2}$$

Inserting 2 in 1 gives:

$$R_{ij}^{\beta} = [E(y_{i1}^{*}) - \alpha_{i1}E(\theta_1)] - [E(y_{j1}^{*}) - \alpha_{j1}E(\theta_1)] - [E(y_{i2}^{*}) - \alpha_{i2}E(\theta_2)] + [E(y_{j2}^{*}) - \alpha_{j2}E(\theta_2)] \tag{3}$$

$$= E(y_{i1}^{*}) - E(y_{j1}^{*}) - E(y_{i2}^{*}) + E(y_{j2}^{*}) + E(\theta_1)(-\alpha_{i1} + \alpha_{j1}) + E(\theta_2)(\alpha_{i2} - \alpha_{j2}) \tag{4}$$

Model identification in terms of the item loadings α_{ig} can arbitrarily be chosen. In order to derive the consequences of identification restrictions of item slopes and variances on relative DIF in item intercepts, we consider two identification restrictions: Let identification A be $\alpha_{i1}^{A} = \alpha_{i2}^{A}$ and $var(\theta_1) = 1$. Let identification B be $\alpha_{j1}^{B} = \alpha_{j2}^{B}$ and $var(\theta_1) = 1$. While the slopes in group 1 are unaffected by the identification constraint, the slopes in group 2 from identification B can be derived from those in identification A as follows:

$$\alpha_{i2}^{B} = \alpha_{i2}^{A}\frac{\alpha_{j1}^{A}}{\alpha_{j2}^{A}} \tag{5}$$

$$\alpha_{j2}^{B} = \alpha_{j2}^{A}\frac{\alpha_{j1}^{A}}{\alpha_{j2}^{A}} \ = \alpha_{j1}^{A}. \tag{6}$$

Thus, R_{ij}^{β} for item i and j with identification A is

$$E(y_{i1}^{*}) - E(y_{j1}^{*}) - E(y_{i2}^{*}) + E(y_{j2}^{*}) + E(\theta_1)(-\alpha_{i1} + \alpha_{j1}) + E(\theta_2)(\alpha_{i2}^{A} - \alpha_{j2}^{A}) \tag{7}$$

R_{ij}^{β} for item i and j with identification B is

$$E(y_{i1}^{*} - E(y_{j1}^{*}) - E(y_{i2}^{*}) + E(y_{j2}^{*}) + E(\theta_1)(-\alpha_{i1} + \alpha_{j1}) + E(\theta_2)(\alpha_{i2}^{B} - \alpha_{j2}^{B}) \tag{8}$$

$$= E(y_{i1}^{*}) - E(y_{j1}^{*}) - E(y_{i2}^{*}) + E(y_{j2}^{*}) + E(\theta_1)(-\alpha_{i1} + \alpha_{j1}) + E(\theta_2)\left(\alpha_{i2}^{A}\frac{\alpha_{j1}^{A}}{\alpha_{j2}^{A}} - \alpha_{j2}^{A}\frac{\alpha_{j1}^{A}}{\alpha_{j2}^{A}}\right) \tag{9}$$

The difference between 7 and 9 is

$$E(y_{i1}^{*}) - E(y_{j1}^{*}) - E(y_{i2}^{*}) + E(y_{j2}^{*}) + E(\theta_1)(-\alpha_{i1} + \alpha_{j1}) + E(\theta_2)(\alpha_{i2}^{A} - \alpha_{j2}^{A}) \tag{10}$$

$$-[E(y_{i1}^{*}) - E(y_{j1}^{*}) - E(y_{i2}^{*}) + E(y_{j2}^{*}) + E(\theta_1)(-\alpha_{i1} + \alpha_{j1}) + E(\theta_2)\left(\alpha_{i2}^{A}\frac{\alpha_{j1}^{A}}{\alpha_{j2}^{A}} - \alpha_{j2}^{A}\frac{\alpha_{j1}^{A}}{\alpha_{j2}^{A}}\right)] \tag{11}$$

$$= E(\theta_2)(\alpha_{i2}^{A} - \alpha_{j2}^{A})\left(1 - \frac{\alpha_{j1}^{A}}{\alpha_{j2}^{A}}\right) \tag{12}$$

If this difference is zero, identification in item slopes does not impact relative DIF in item intercepts. This term will be zero only if the true mean of ability in the second group is zero ($E(\theta_2) = 0$), if the slopes of the items are the same within the second group ($\alpha_{i2}^A = \alpha_{j2}^A$) or if there is no relative DIF of item j ($\alpha_{j1}^A = \alpha_{j2}^A$). The first condition can not be assumed to hold in general. The second condition is only fulfilled in Rasch models, but not in 2PL models. The latter is only fulfilled when there is no DIF in item slopes. As none of the condition can generally be assumed to hold in such applications, the relative DIF in item intercepts may be impacted by identification restrictions in item slopes.

R code

```r
# The IRT estimators used come from mirt.
# No data is provided in these examples.
# requested packages:
library(mirt)
library(doParallel)
library(plyr)
library(rlist)
library(Ckmeans.1d.dp)
library(shape)
# There are two main functions:
#   - twoStepThreshold for the 2 groups case to apply the cluster algorithm with various thresholds at once.
#   - plot2PLclusters for plotting the results of one of the above.
########################## Two Groups ##########################
# DIF analysis for 2 groups. Returns a list with the results.
# Prints a duration estimate. Has its own summary class.
res <- twoStepThreshold(
data,                       # observed data (items answers only)
group,                      # grouping variable
aThreshold = 0.3,           # alpha threshold (can be a scalar or vector)
bThreshold = c(0.5, 0.6), # beta threshold (can be a scalar or vector)
nCPUs = 4)                  # (maximum) number of processors to be used in parallel in the second step (the first step is single core only)
# An overview over the final cluster results for varying threshold combinations:
summary(res)
# A detailed look at the returned list. It is generally structured in the way: list(alphaThresholds = list(betaThresholds))
str(res, 2)

########################## Plotting Results ##########################
# Plot results from 2PL clustering. Saves plot straight to a file.
plot2PLclusters(
res,                          # the result list from twoStepThreshold
aThresh=0.3,                  # alpha threshold for which plot is drawn
bThresh=0.5,                  # beta threshold for which plot is drawn
tickWidth=0.2,                # spacing of ticks on x-axis
fileName="example 2PL plot") # beginning of file name, thresholds are added automatically

########### Estimate 2PL Model for Specific Cluster Solution ############
cluster <- res$`aThresh=0.3`$`bThresh=0.5`$cluster2ndStep
anchor <- which(cluster == 1)  # Get positions of the anchor items. Here, cluster 1 is used as an example.

mod <- mirt:::multipleGroup(data,
group = group,
invariance = c("free_means", "free_var", names(data)[anchor]), # the invariant items
model = 1,
itemtype = "2PL")
coef(mod, simplify=T)  # show item parameters and latent moments

################## Functions ##############
# 1)  2step algorithm for 2PL model for groups.
# Yields a list with the resulting cluster memberships and parameter estimates per step.
twoStepThreshold <- function(
Dat,
```

```
52  groups,
53  aThresholds,     # can be a single value or a vector of values (i.e., a sequence)
54  bThresholds,     # can be a single value or a vector of values (i.e., a sequence)
55  nCPUs = 1
56  ) {     # (maximum) number of processors to be used in 2nd step
57
58  threshL <- list()    #  intialize level 2 list
59  if (length(unique(groups)) > 2) stop("Group number is wrong. Exactly 2 groups are
        needed.")
60
61  # initial model
62  cat("Estimating initial model...")
63  t0 <- proc.time()
64  mod1 <- multipleGroup(Dat,
65  model = 1,
66  itemtype = "2PL",
67  #method = "MHRM",
68  group = groups,
69  invariance = character(),
70  SE = T, verbose=F)
71  t1 <- proc.time()
72  tDiff <- as.numeric(t1 - t0)[3]
73
74  params1 <- getItemParams(mod1, SE=F)
75
76  if (min(params1[c(2, 4)]) < 0) {
77  "Negative loadings. Cannot continue."
78  stop()
79  }
80  ##  2step approach
81  #  1st step: alphas
82
83  for (aThresh in aThresholds) {  #  main loop for various a-thresholds
84  dR_alpha <- drids2PL(mod1)$dR_alpha
85  cluster1stStep <- kMeansThresh(dR_alpha[, 1], aThresh)
86
87  cat("\r", "Current alpha threshold =", aThresh, "with approx. duration =",
88  ceiling(ceiling(max(cluster1stStep) / nCPUs)*tDiff/60), "minutes")
89
90  #  2nd step: betas
91  cluster2ndStepL <- list(cluster1stStep = cluster1stStep, model1stStep = list(params
        = params1,
92  output = capture.output(mod1)))
93  for (bThresh in bThresholds)  cluster2ndStepL[[paste0("bThresh=", bThresh)]][["
        cluster2ndStep"]] <- vector(length = length(cluster1stStep))    #  initialize
        lower storage list
94  cluster1stStep_1 <- unname(which(table(cluster1stStep) == 1))           #  check
        for 1-item alpha clusters...
95  cluster1stStep_for2ndStep <- unname(which(table(cluster1stStep) > 1))   #  ...and
        here all clusters with more than 1 item
96
97  # parallized 2nd step estimation (only for alpha clusters with at least 2 items)
98  cl <- makeCluster(nCPUs)   # initialize parallization
99  registerDoParallel(cl)
100
101 mod2L <- foreach(curr1step = cluster1stStep_for2ndStep, .inorder=T, .combine=append
        ,
102 .packages=c("mirt","Ckmeans.1d.dp"), .errorhandling='stop') %dopar% {
103
104 currCluster <- which(cluster1stStep == curr1step)   # get current items
```

```
105 anchorMod <- paste0("F1 = 1-", length(cluster1stStep),"
106 CONSTRAINB = (", paste(curr1step, collapse=","), ", a1), (", sample(curr1step, 1),
      ", d)") #  constrain item of current alpha cluster and a random intercept from
    the current cluster
107 mod2 <- multipleGroup(Dat,
108 model = anchorMod,
109 itemtype = "2PL",
110 #method = "MHRM",
111 group = groups,
112 invariance = c("free_var"),  #"free_means",
113 SE = T, verbose=F)
114 }
115 stopCluster(cl) # cleanly exit parallel threads
116
117 if (length(cluster1stStep_for2ndStep) == 1) mod2L <- list(mod2L) # achieve list
    structure even when there is only 1 entry
118
119 for (i in cluster1stStep_for2ndStep) {
120 curr1step <- which(cluster1stStep == i)   # get current items
121
122 params2 <- getItemParams(mod2L[[which(cluster1stStep_for2ndStep == i)]], SE=F)
123 dR_beta <- drids2PL(mod2L[[which(cluster1stStep_for2ndStep == i)]])$dR_beta
124 cluster2ndStepL[["model2ndStep"]][[paste0("1st cluster=", i)]] <- list(params=
    params2,
125 output=capture.output(mod2L[[which(cluster1stStep_for2ndStep == i)]]))
126
127 for (bThresh in bThresholds) {
128 curr2step <- kMeansThresh(dR_beta[curr1step, 1], bThresh)
129 cluster2ndStepL[[paste0("bThresh=", bThresh)]][["cluster2ndStep"]][curr1step] <-
    curr2step + i*100   #  i*100 produces unique cluster labels
130 historySteps <- rep(NA, length(cluster1stStep))  #  save history
131 historySteps[curr1step] <- curr2step
132 cluster2ndStepL[[paste0("bThresh=", bThresh)]][[paste0("history: 1st cluster=", i)
    ]] <- historySteps
133 }
134 }
135
136 for (i in cluster1stStep_1) {
137 for (bThresh in bThresholds) {
138 cluster2ndStepL[[paste0("bThresh=", bThresh)]][["cluster2ndStep"]][cluster1stStep==
    i] <- rep(i, 1)  #  for 1-item clusters: give cluster codes that will not occur
    otherwise (below 100)
139
140 }
141 }
142 # unify cluster labels
143 for (bThresh in bThresholds) cluster2ndStepL[[paste0("bThresh=", bThresh)]][["
    cluster2ndStep"]] <- as.integer(factor(cluster2ndStepL[[paste0("bThresh=",
    bThresh)]][["cluster2ndStep"]]))
144 ##  end 2step approach
145
146 threshL[[paste0("aThresh=", aThresh)]] <- cluster2ndStepL
147 }
148
149 class(threshL) <- "twoStepClass"   # give the result list a class attribute in
    order to be able to apply a customized summary-function
150 return(threshL)
151 }
152
153 summary.twoStepClass <- function(threshL,
```

```
inRows = F
) {
stopifnot(inherits(threshL, "twoStepClass"))
Res <- data.frame()
i <- 0
nItems <- length(threshL[[c(1, 3, 1)]])

if (inRows == F) {
for (aThresh in 1:length(threshL)) {
for (bThresh in 1:(length(threshL[[1]]) - 3)) {
i <- i + 1
Res[1, i] <- sub(".......=", "", names(threshL[aThresh]))                    # save a-threshold
Res[2, i] <- sub(".......=", "", names(threshL[[aThresh]][bThresh + 2]))     # save b-threshold
Res[3, i] <- max(t(t(threshL[[c(aThresh, (bThresh + 2), 1)]])))              # save number of clusters
Res[4, i] <- ""
Res[5:(nItems + 4), i] <- t(t(threshL[[c(aThresh, (bThresh + 2), 1)]]))      # save cluster code
}
}
rownames(Res) <- c("aThresh", "bThresh", "nClusters", "", paste0("i", 1:nItems))
colnames(Res) <- NULL
cat("#### Final clusters found in 2step algorithm ####", "\n")
} else {
for (aThresh in 1:length(threshL)) {
for (bThresh in 1:(length(threshL[[1]]) - 3)) {
i <- i + 1
Res <- rbind(Res, as.numeric(c(sub(".......=", "", names(threshL[aThresh])),
                    # save a-threshold
sub(".......=", "", names(threshL[[aThresh]][bThresh + 2])), # save b-threshold
max(t(t(threshL[[c(aThresh, (bThresh + 2), 1)]]))),           # save number of clusters
threshL[[c(aThresh, (bThresh + 2), 1)]])))                   # save cluster code
}
}
colnames(Res) <- c("aThresh", "bThresh", "nClusters", paste0("i", 1:nItems))
cat("#### Final clusters found in 2step algorithm ####", "\n", "\n")
}
return(Res)
}


## 2) Function for plotting the results. Saves plot straight to a file.
plot2PLclusters <- function(
res,                                # result list from twoStepThreshold function
aThresh,                            # alpha threshold for which plot is drawn
bThresh,                            # beta threshold for which plot is drawn
fileName = "2PL clustering plot", # beginning of file name, to allow custom file names
tickWidth = 0.2                     # length of tick interval in DRID plots
) {
#  get clusterings
cluster1stStep <- res[[paste0("aThresh=", aThresh)]]$cluster1stStep
cluster2ndStep <- res[[c(paste0("aThresh=", aThresh), paste0("bThresh=", bThresh))]]$cluster2ndStep

cluster1stStep_1 <- unname(which(table(cluster1stStep) == 1))           #  check for 1-item alpha clusters...
```

```
cluster1stStep_for2ndStep <- unname(which(table(cluster1stStep) > 1))   #  ...and
    here all clusters with more than 1 item
clusCount <- length(cluster1stStep_for2ndStep)

#  get relative DIFs
alphas <- res[[paste0("aThresh=", aThresh)]]$model1stStep$params
r1 <- outer(log(alphas[, 1]), log(alphas[, 1]), "-")
r2 <- outer(log(alphas[, 3]), log(alphas[, 3]), "-")
dR_alpha <- (r1 - r2)[, 1]

dR_beta <- vector(length=length(cluster1stStep))
#  getting beta DRIDs for non 1-item clusters
for (i in cluster1stStep_for2ndStep) {
betasCurrent <- res[[paste0("aThresh=", aThresh)]]$model2ndStep[[paste0("1st_
    cluster=", i)]]$params
r1 <- outer(betasCurrent[, 2], betasCurrent[, 2], "-")
r2 <- outer(betasCurrent[, 4], betasCurrent[, 4], "-")
dR_beta[cluster1stStep == i] <- (r1 - r2)[cluster1stStep == i, 1]
}

#  calculate plot widths
totalRange <- 0
previousRange <- 0
rangeL <- list() #  store cumulative ranges of the single 2nd step plots to
    calculate their individual plot widths later
for (pos in cluster1stStep_for2ndStep) {
minD <- min(dR_beta[cluster1stStep == pos] - min(dR_beta[cluster1stStep == pos]))
maxD <- max(dR_beta[cluster1stStep == pos] - min(dR_beta[cluster1stStep == pos]))
tickX <- seq(plyr:::round_any(minD, tickWidth, floor), plyr:::round_any(maxD,
    tickWidth, ceiling),
by = tickWidth)
rangeL[[pos]] <- previousRange
totalRange <- totalRange + max(tickX) - min(tickX) + tickWidth
previousRange <- previousRange + max(tickX) - min(tickX) + tickWidth
}

##  plotting
png(file=paste0(fileName, "_", aThresh, "_", bThresh, ".png"),
width=200*(totalRange / tickWidth), height=1500, res=300, family="serif")
plot.new()
par(mai=c(0, 0, 0, 0))
plot(0, type="n", axes=F, xaxs="i", #  empty plot, setting up plotting area
ylim=c(0, 3),
xlim = c(-totalRange/40, totalRange))

# 1st step
cols1stStep <- c("red", "blue", "palevioletred", "skyblue1",   #  12 colors for 1st
     step (red and blueish)
"firebrick3", "steelblue3","coral1", "dodgerblue2",
"orangered2", "navy", "deeppink4", "cyan")
if (length(cluster1stStep_for2ndStep) > 12) print("Color_coding_in_step_1_has_
    failed_due_to_high_cluster_number.")
cols <- vector()
cols[cluster1stStep_for2ndStep] <- cols1stStep[as.integer(factor(cluster1stStep_
    for2ndStep))]  # the as.integer(factor(..)) construct lets R start with the
    first color
cols[cluster1stStep_1] <- "black"

text(labels=expression("1"^"st"*"_step:_"*alpha*"-clusters"), x=0, y=2.9, cex=1,
    pos=4) #  1st heading
```

```
256 plotClusters(dR_alpha - min(dR_alpha), #  shift values to start with 0
257 x = 0, y = 2,
258 tickWidth = tickWidth,
259 clustering = cluster1stStep,
260 broken = rep(0, length(cluster1stStep)),
261 clusterColors = cols)
262 text(labels=expression("Relative DIF in "*alpha), x=(max(dR_alpha - min(dR_alpha))
    / 2), y=2.35, cex=0.7)
263
264 #  2nd step
265 cols2ndStep <- rep(c("purple", "olivedrab3", "darkorange1", #  color coding in 9
    colors, three times over (full line, once broken, twice broken)
266 "magenta1", "lawngreen", "brown",
267 "plum", "seagreen", "burlywood"), 3)
268 brokenS <- c(rep("0", 9), rep("1", 9), rep("2", 9))
269 if (length(unique(cluster2ndStep)) > 27) print("Color coding in step 2 has failed
    due to high cluster number.")
270
271 thirdPlotCols <- vector(length=length(unique(cluster2ndStep))) #  to contain the
    color coding of the 2nd step to print a third plot for all items
272 thirdPlotBroken <- vector(length=length(unique(cluster2ndStep))) #  ... and broken
    lines coding likewise
273
274 text(labels=expression("2"^"nd"*" step: "*beta*"-clusters"), x=0, y=1.9, cex=1, pos
    =4) #  2nd heading
275
276 for (pos in cluster1stStep_for2ndStep) {
277 currClus <- sort(unique(cluster2ndStep[cluster1stStep == pos]))
278
279 plotClusters(dR_beta[cluster1stStep == pos] - min(dR_beta[cluster1stStep == pos]),
    #  shift values to be greater than 0
280 x = rangeL[[pos]], y = 1,
281 tickWidth = tickWidth,
282 clustering = cluster2ndStep[cluster1stStep == pos],
283 axisCol = cols1stStep[which(unique(cluster1stStep_for2ndStep) == pos)],
284 clusterColors = cols2ndStep[currClus],
285 broken = brokenS[currClus])
286 text(labels=expression("Relative DIF in "*beta), cex=0.7,
287 x=rangeL[[pos]] + (max(dR_beta[cluster1stStep == pos] - min(dR_beta[cluster1stStep
    == pos])) / 2), y=1.35)
288 thirdPlotCols[currClus] <- cols2ndStep[currClus]
289 thirdPlotBroken[currClus] <- brokenS[currClus]
290 }
291 thirdPlotCols[thirdPlotCols == "FALSE"] <- "black"   #  insert values for 1-item
    clusters
292 thirdPlotBroken[thirdPlotBroken == "FALSE"] <- "0"
293
294 #  3rd: plot summary for all items
295 text(labels="Cluster summary", x=0, y=0.9, cex=1, pos=4)
296 plotClusters(seq(0, totalRange/2, by=(totalRange/2)/(length(cluster1stStep)-1)),
297 x = 0, y = 0,
298 tickWidth = tickWidth,
299 clustering = cluster2ndStep,
300 clusterColors = thirdPlotCols,
301 broken = thirdPlotBroken,
302 thirdPlot = T)
303 text(labels="Item number", x=totalRange/4, y=0.3, cex=0.7)
304
305 #  plot legend
306 shape:::Arrows(totalRange*0.55, 0.66, totalRange*0.55+tickWidth, 0.66,
```

```
307 arr.length=0.1, code=3, arr.adj = 1, lwd=0.8)
308 shape:::Arrows(totalRange*0.55, 0.66, totalRange*0.55+tickWidth, 0.66,
309 cex=0.3, code=3, arr.adj = 1, arr.type="T", lwd=0.8)
310 text(totalRange*0.55+tickWidth, 0.66, paste(tickWidth, "logits"), pos=4, cex=0.7)
311 text(totalRange*0.55, 0.55, bquote(.(max(cluster1stStep))~"clusters in 1"^"st"~"
    step"), pos=4, cex=0.7)
312 text(totalRange*0.55, 0.46, bquote(.(max(cluster2ndStep))~"clusters in 2"^"nd"~"
    step"), pos=4, cex=0.7)
313 text(totalRange*0.55, 0.37, bquote(alpha*"-threshold = "*.(aThresh)), pos=4, cex
    =0.7)
314 text(totalRange*0.55, 0.26, bquote(beta*"-threshold = "*.(bThresh)), pos=4, cex
    =0.7)
315
316 dev.off()
317 }
318
319 ################# Auxilary functions ##########################
320 ## plots 1d clustering data
321 plotClusters <- function(
322 Dat,                  # input data
323 axisCol = "black",
324 x = NULL,             # x position
325 y = NULL,             # y position
326 tickLabels = F,       # switch tick labels on and off
327 clustering = NULL,    # numeric[n]: cluster a point in d belongs to
328 broken = NULL,        # numeric[0, 1], coding, which clusters should be displayed
    as broken lines for clarity
329 clusterColors = NULL, # character[n]: colors for the clusters
330 thirdPlot=FALSE,      # changes somee properties, if this is the third part of the
    plot
331 tickWidth=tickWidth
332 ) {
333
334 tickX <- seq(round_any(min(Dat), tickWidth, floor), round_any(max(Dat), tickWidth,
    ceiling),
335 by = tickWidth)
336 if (thirdPlot) {
337 tickX <- Dat
338 text(x+tickX, y+0.4, labels=c(1:length(Dat)), cex=0.7)
339 }
340 clustering <- as.integer(factor(clustering)) #  getting cluster names to starting
    with 1
341
342 #  draw lines in plot
343 for (i in unique(clustering)) {
344 segments(x+Dat[clustering == i], y+0.6, x+Dat[clustering == i], y+0.7, col=
    clusterColors[i], lwd=2)
345 if (broken[i] == "1") {
346 segments(x+Dat[clustering == i], y+0.64, x+Dat[clustering == i], y+0.66, col="white
    ", lwd=2, lend=1)
347 }
348 if (broken[i] == "2") {
349 segments(x+Dat[clustering == i], y+0.6233, x+Dat[clustering == i], y+0.6433, col="
    white", lwd=2, lend=1)
350 segments(x+Dat[clustering == i], y+0.6566, x+Dat[clustering == i], y+0.6766, col="
    white", lwd=2, lend=1)
351 }
352 }
353 #  draw axis
354 lines(c(x+min(tickX), x+max(tickX)),
```

```
355 c(y+0.5, y+0.5),
356 col = axisCol, lwd = 2, lend=2)
357 segments(x+tickX, y+0.47, x+tickX, y+0.53, col = axisCol, lend=1)
358 }
359
360 ## calculate relative DIFs (2 groups)
361 drids2PL <- function(mod # mirt-model object (two groups)
362 ){
363 itemPars <- getItemParams(mod, SE = FALSE)
364 r1 <- outer(itemPars[, 1], itemPars[, 1], "-")
365 r2 <- outer(itemPars[, 3], itemPars[, 3], "-")
366 dR_beta <- r1 - r2
367 colnames(dR_beta) <-  extract.mirt(mod, "itemnames")
368
369 r1 <- outer(log(itemPars[, 2]), log(itemPars[, 2]), "-")
370 r2 <- outer(log(itemPars[, 4]), log(itemPars[, 4]), "-")
371 dR_alpha <- r1 - r2
372 colnames(dR_alpha) <- extract.mirt(mod, "itemnames")
373 res <- list(dR_alpha = dR_alpha, dR_beta = dR_beta)
374 return(res)
375 }
376
377 ## retrieve parameters from mirt in a digestable way
378 getItemParams <- function(mod,                    # mirt-model
379 params = c("d", "a1"), # character[i]: names of the item parameters to be extracted
      (in general: d/a1/g/u)
380 SE = TRUE,             # logical: should parameter standard errors be reported as
     well?
381 addVar = NULL,         # numeric[k]: add an additional/ multiple variables to the
     output (for ecample to compare simulated values), use cbind() to give them
     useful names
382 roundTo = 5            # numeric[1]: decimals, the values should be rounded to
383 #-> Q4
384 ){
385
386 containsSE <- !is.na(vcov(mod)[1,1]) # does the model include SEs? (estimated with
     SE = TRUE?)
387 if (!containsSE && SE) {
388 warning("Standard␣errors␣were␣requested␣but␣not␣estimated.␣They␣will␣not␣be␣
     reported...")
389 SE <- FALSE
390 }
391 itNames <- extract.mirt(mod, "itemnames")
392 grpNames <- extract.mirt(mod, "groupNames")
393 out <- data.frame(matrix(NA, nrow = length(itNames), ncol = 1))
394 cfg <- coef(mod, printSE = TRUE)
395 res <- data.frame(t(data.frame(cfg)))
396 for (g in grpNames){
397 if (length(grpNames) > 1) {
398 grpIdx <- grep(g, rownames(res))
399 } else {
400 grpIdx <- 1:nrow(res)
401 }
402 s <- strsplit(rownames(res),".", fixed = TRUE)
403 parVec <- sapply(s,function(x) x[length(x)])
404 for (p in params){
405 parIdx <- which(parVec == p)
406 curr <- res[intersect(parIdx,grpIdx),]
407 if (!is.numeric(curr)) curr <- curr[,"par"]
408 if (length(grpNames) == 1) {
```

```
409 colName <- p
410 } else {
411 colName <- paste(g, p, sep = ".")
412 }
413 out[colName] <- curr
414 if (SE && containsSE){
415 curr <- res[intersect(parIdx,grpIdx),]
416 out[paste(colName, "SE",sep = ".")] <- curr[, "SE"]
417 }
418 }
419 }
420 out <- out[-1]
421 rownames(out) <- itNames
422 if (!is.null(addVar)) out <- cbind(out, addVar)
423 return(as.data.frame(round(out, roundTo)))
424 }
425
426 ## apply k-means clustering with a threshold
427 kMeansThresh <- function(drids,          # relative DIF-values for the items (taken
      from delta-R matrix)
428 thresh = NULL  # threshhold as a stopping rule. Threshold is maximum cluster width
      on logit scale
429 ){
430 k <- 1
431 currRange <- max(drids) - min(drids)
432 while(max(currRange) > thresh & k < length(drids)) {
433 k <- k + 1
434 clusCode <- Ckmeans.1d.dp(drids, k=k)$cluster
435 currRange <- NULL
436 for(i in 1:max(clusCode)) {
437 clusCode2 <- clusCode      # helper copy
438 clusCode2[clusCode2 != i] <- NA          # make picking vector
439 clusCode2[clusCode2 == i] <- 1
440 currRange[i] <- max(clusCode2*drids, na.rm=T) - min(clusCode2*drids, na.rm=T)    #
      get current range
441 }
442 }
443 res <- Ckmeans.1d.dp(drids, k=k)$cluster
444 res
445 }
```

Design of simulation study

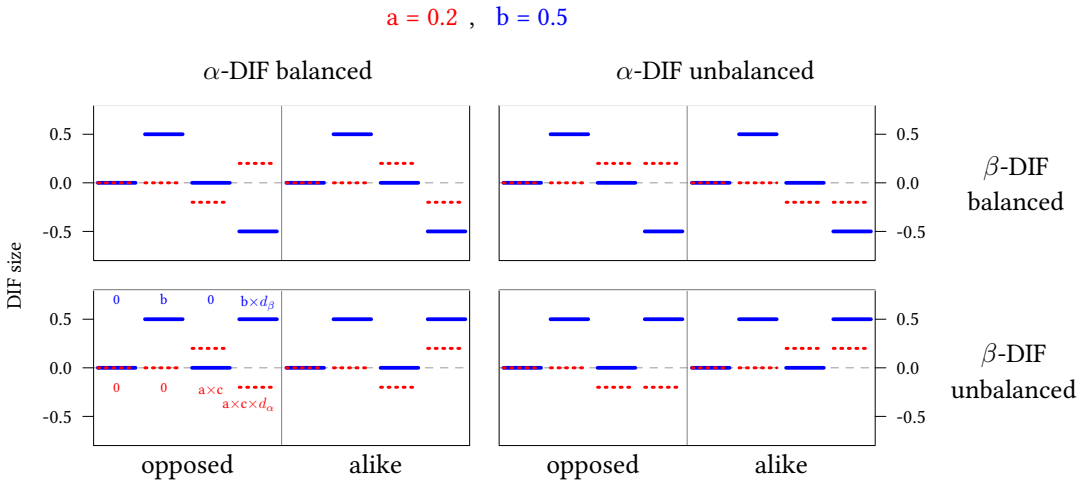

Figure S1: Data generating conditions in the simulation study with five simulated factors: α-DIF size, β-DIF size, balancedness of α-DIF and β-DIF, and mutual direction of α-DIF and β-DIF.

Results compared across DIF-sizes

α balanced, β balanced, alike

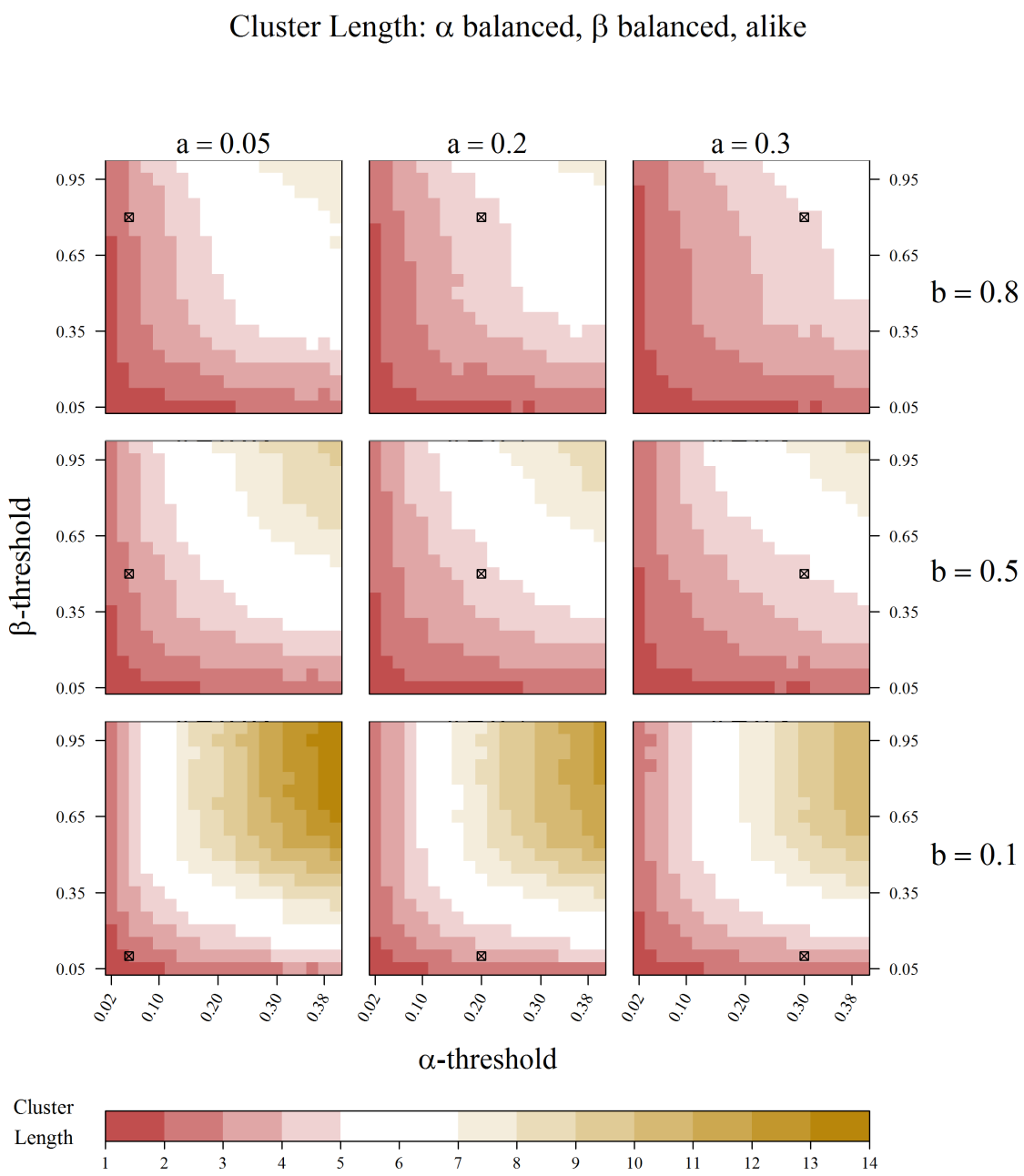

Figure S2: Cluster length for the best cluster in the condition of balanced α-DIF, balanced β-DIF, and same direction of DIF.

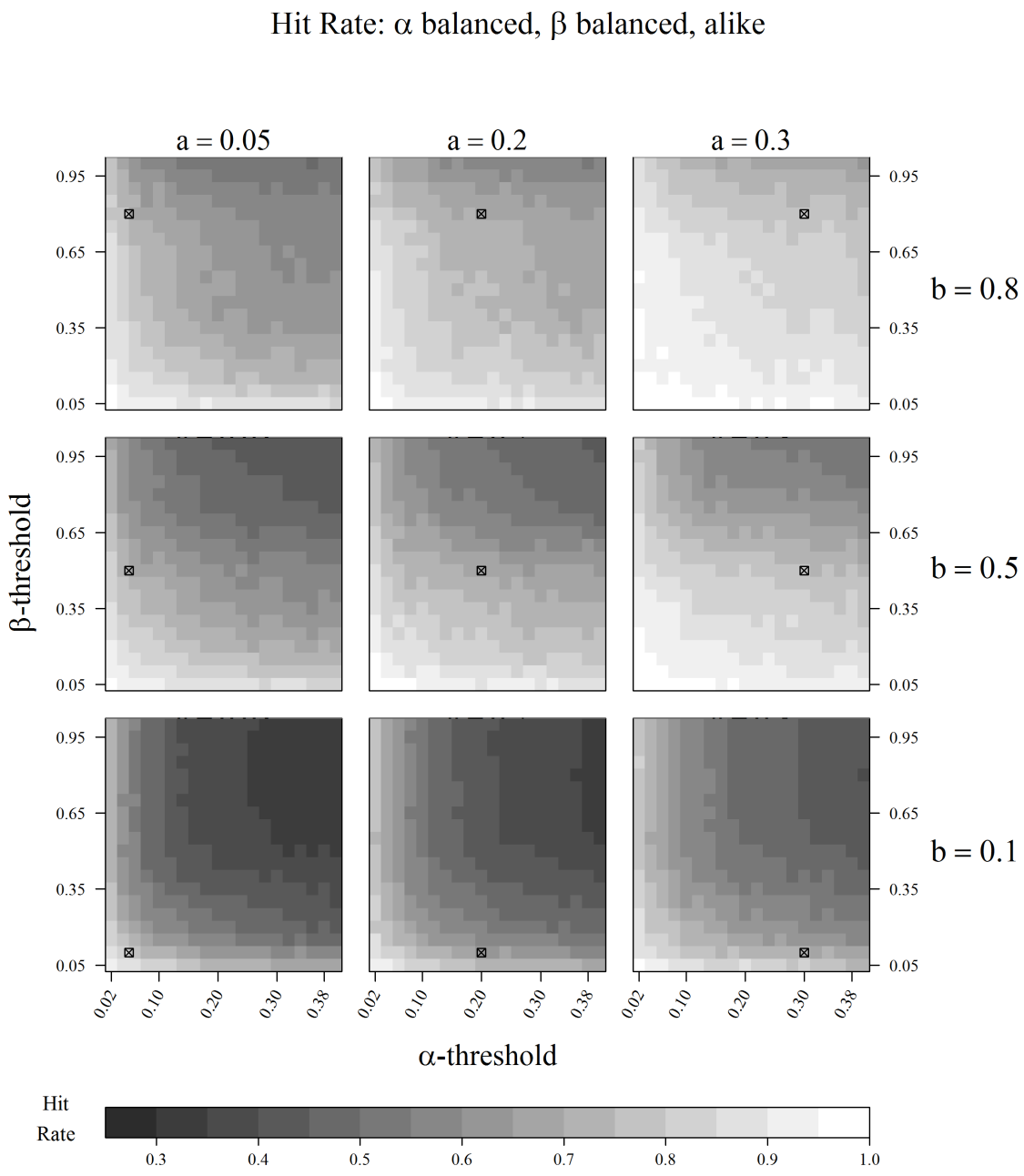

Figure S3: Hit rate for the best cluster in the condition of balanced α-DIF, balanced β-DIF, and same direction of DIF.

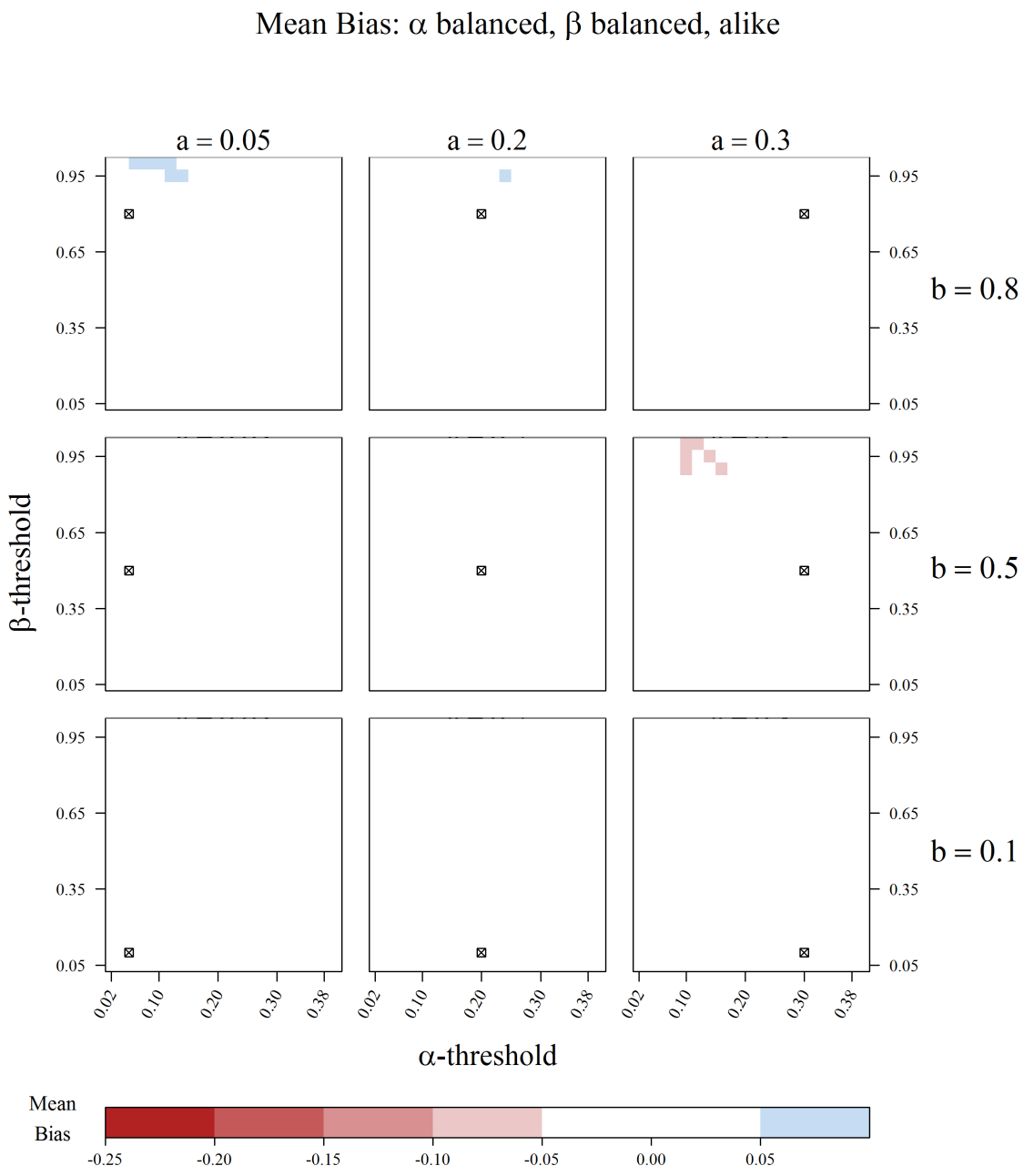

Figure S4: Bias in estimated mean difference for the best cluster in the condition of balanced α-DIF, balanced β-DIF, and same direction of DIF.

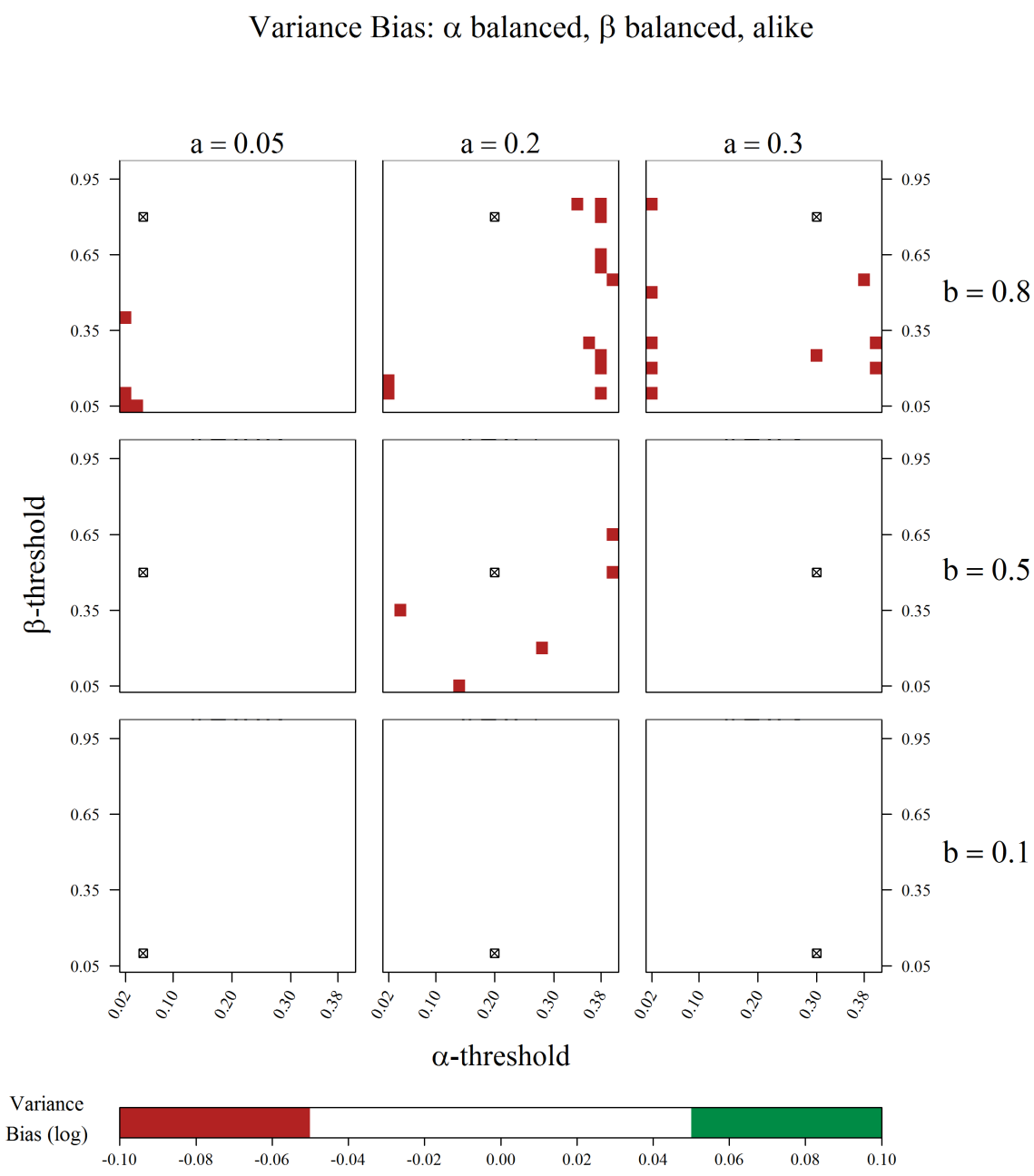

Figure S5: Bias in estimated relation of variances for the best cluster in the condition of balanced α-DIF, balanced β-DIF, and same direction of DIF.

α balanced, β balanced, opposed

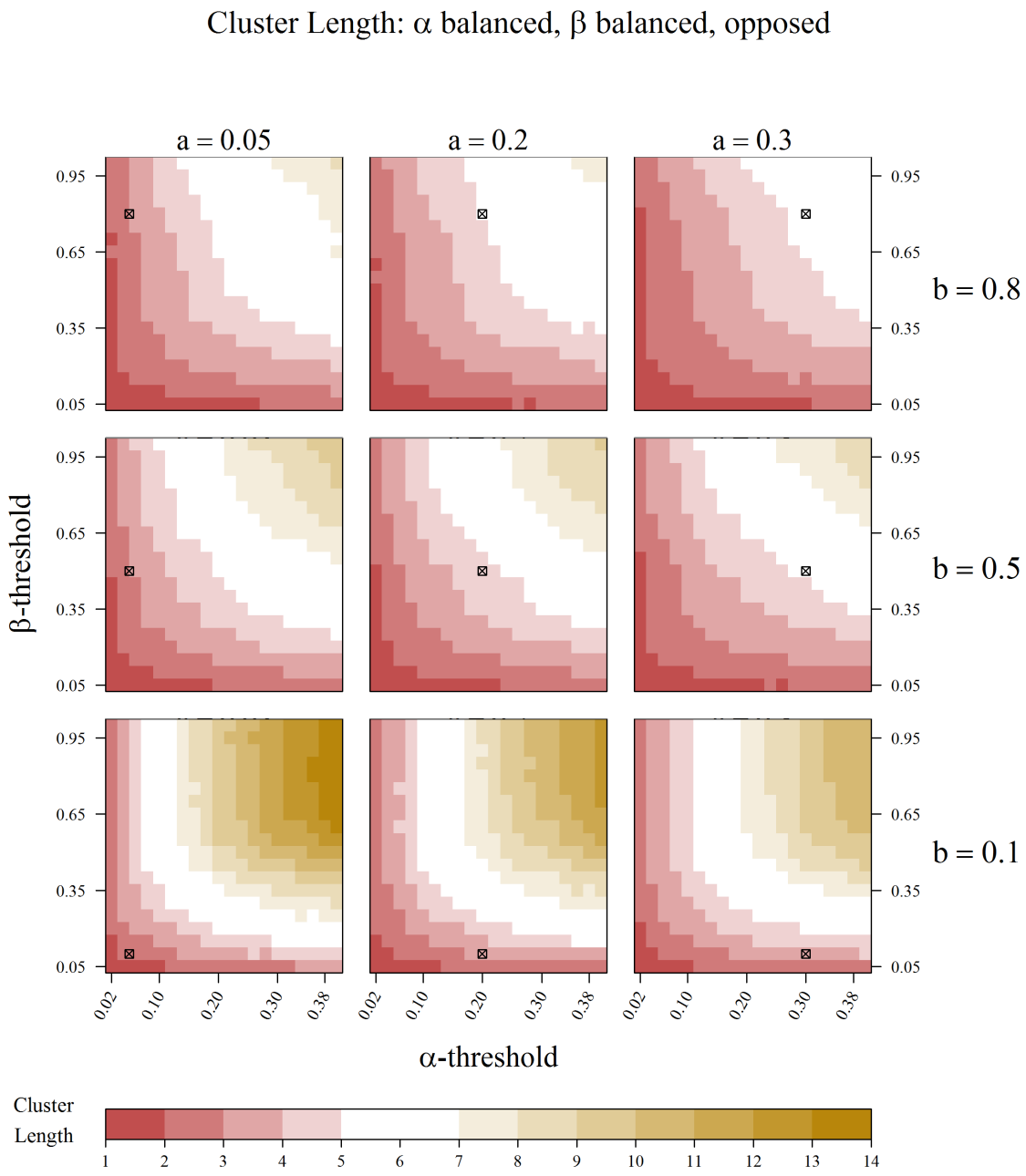

Figure S6: Cluster length for the best cluster in the condition of balanced α-DIF, balanced β-DIF, and opposed direction of DIF.

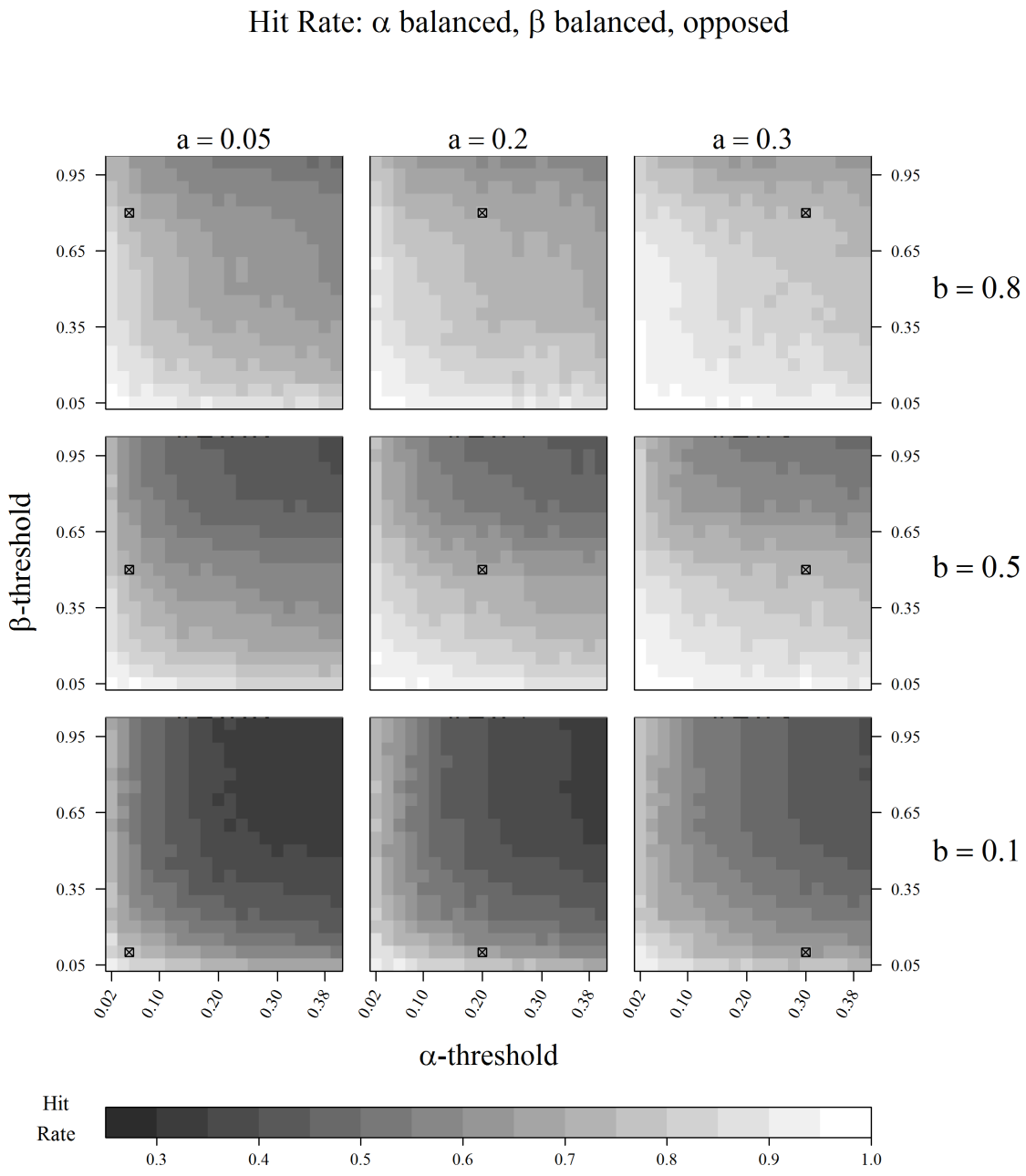

Figure S7: Hit rate for the best cluster in the condition of balanced α-DIF, balanced β-DIF, and opposed direction of DIF.

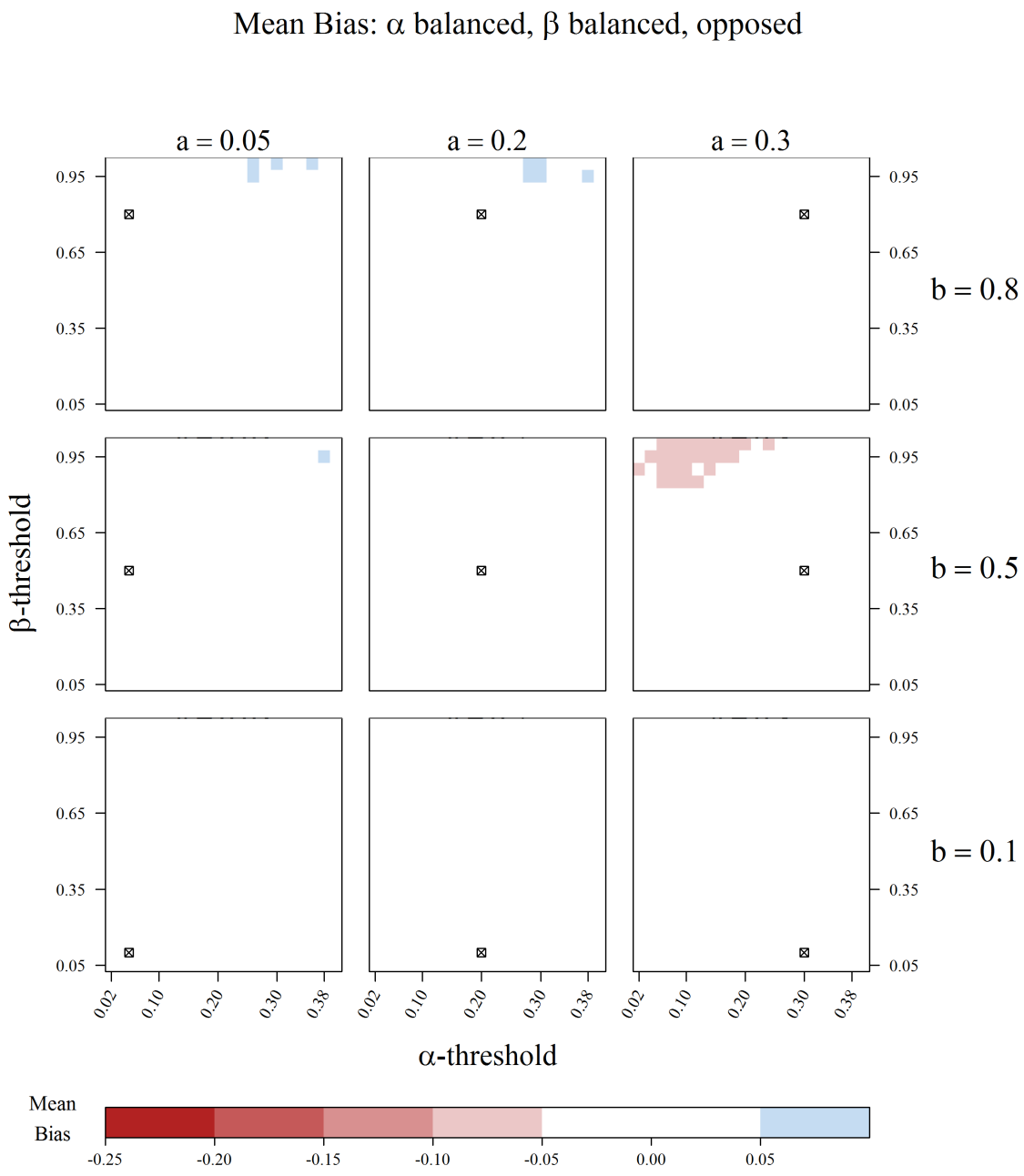

Figure S8: Bias in estimated mean difference for the best cluster in the condition of balanced α-DIF, balanced β-DIF, and opposed direction of DIF.

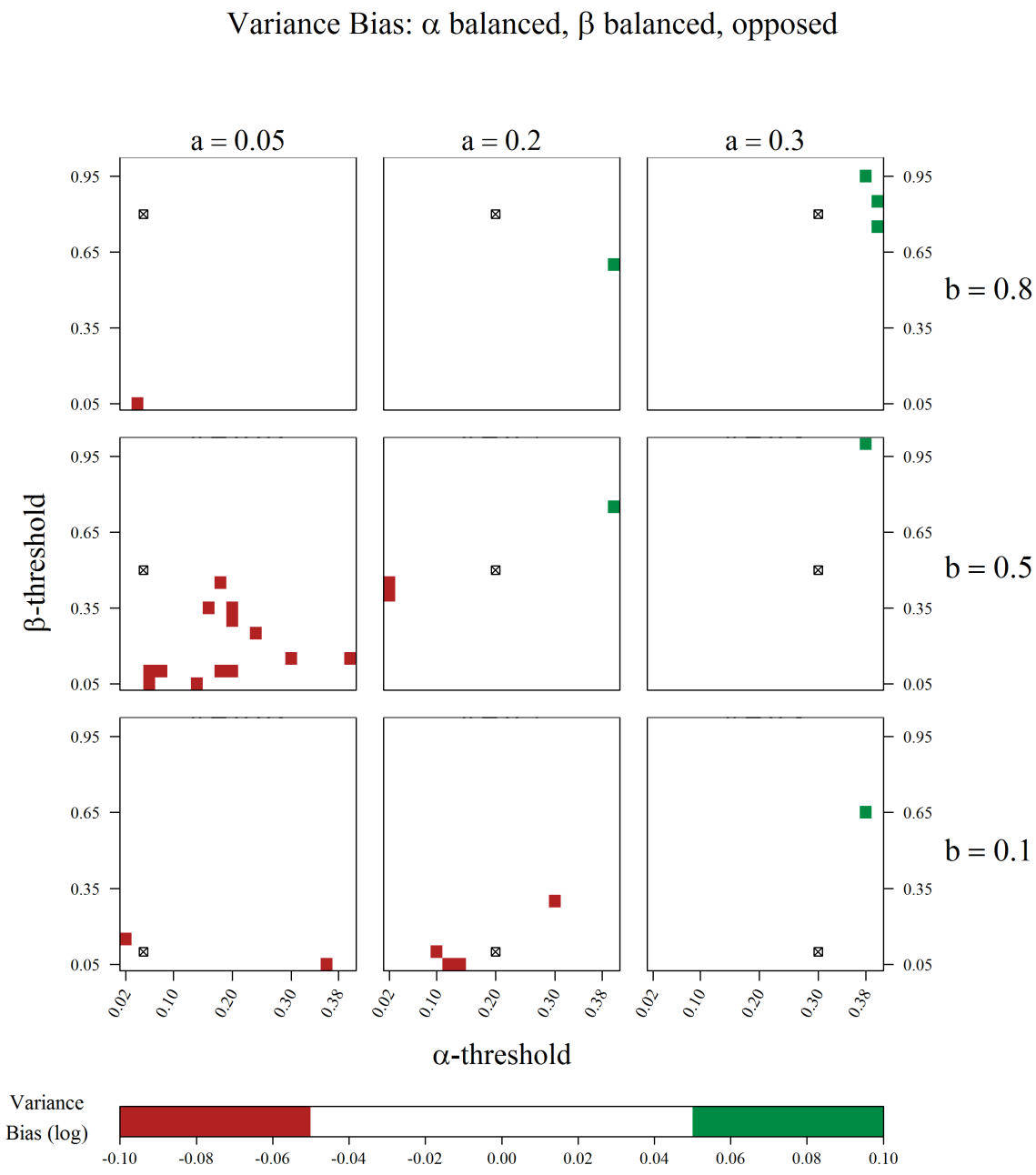

Figure S9: Bias in estimated relation of variances for the best cluster in the condition of balanced α-DIF, balanced β-DIF, and opposed direction of DIF.

α balanced, β unbalanced, alike

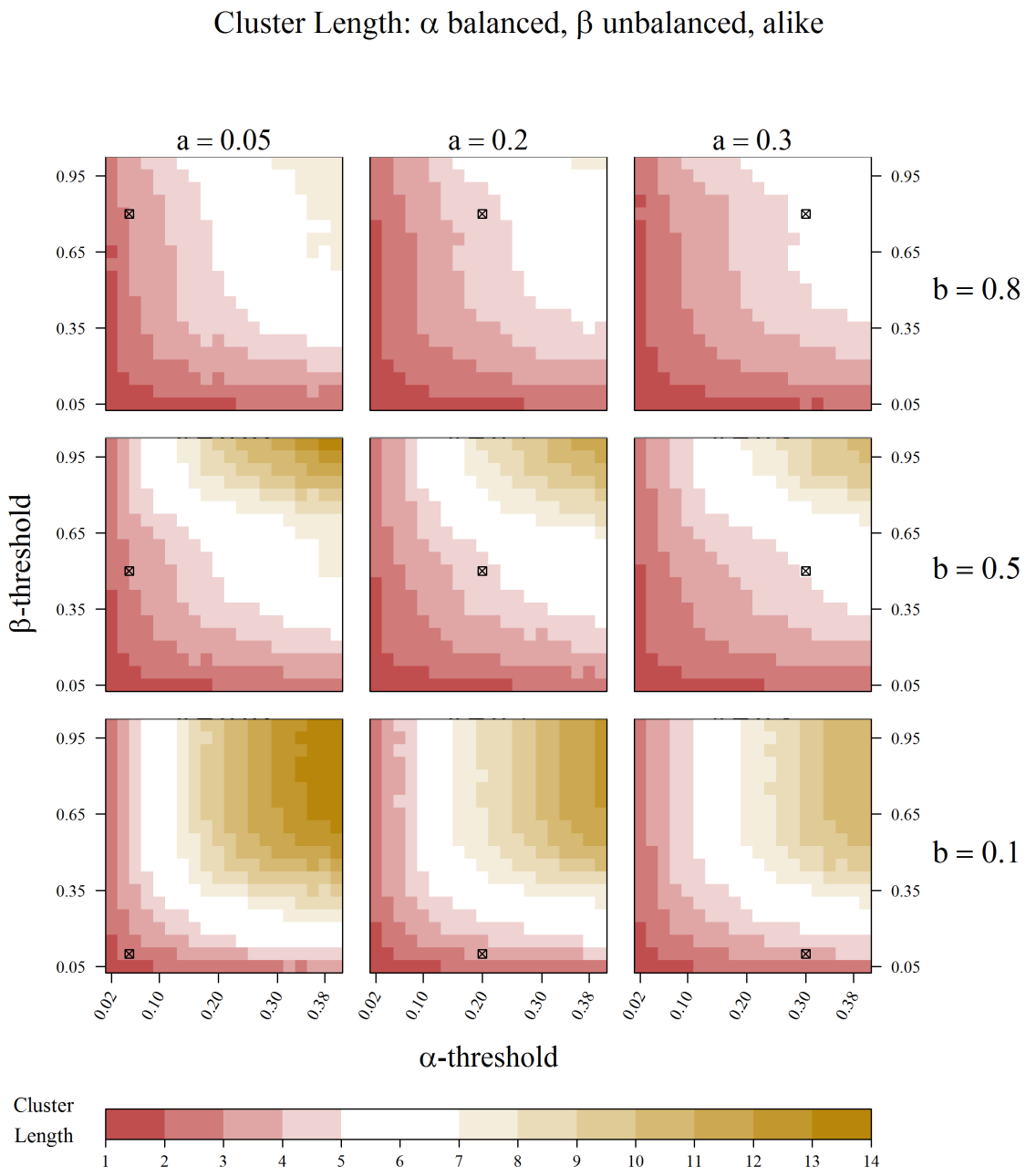

Figure S10: Cluster length for the best cluster in the condition of balanced α-DIF, unbalanced β-DIF, and same direction of DIF.

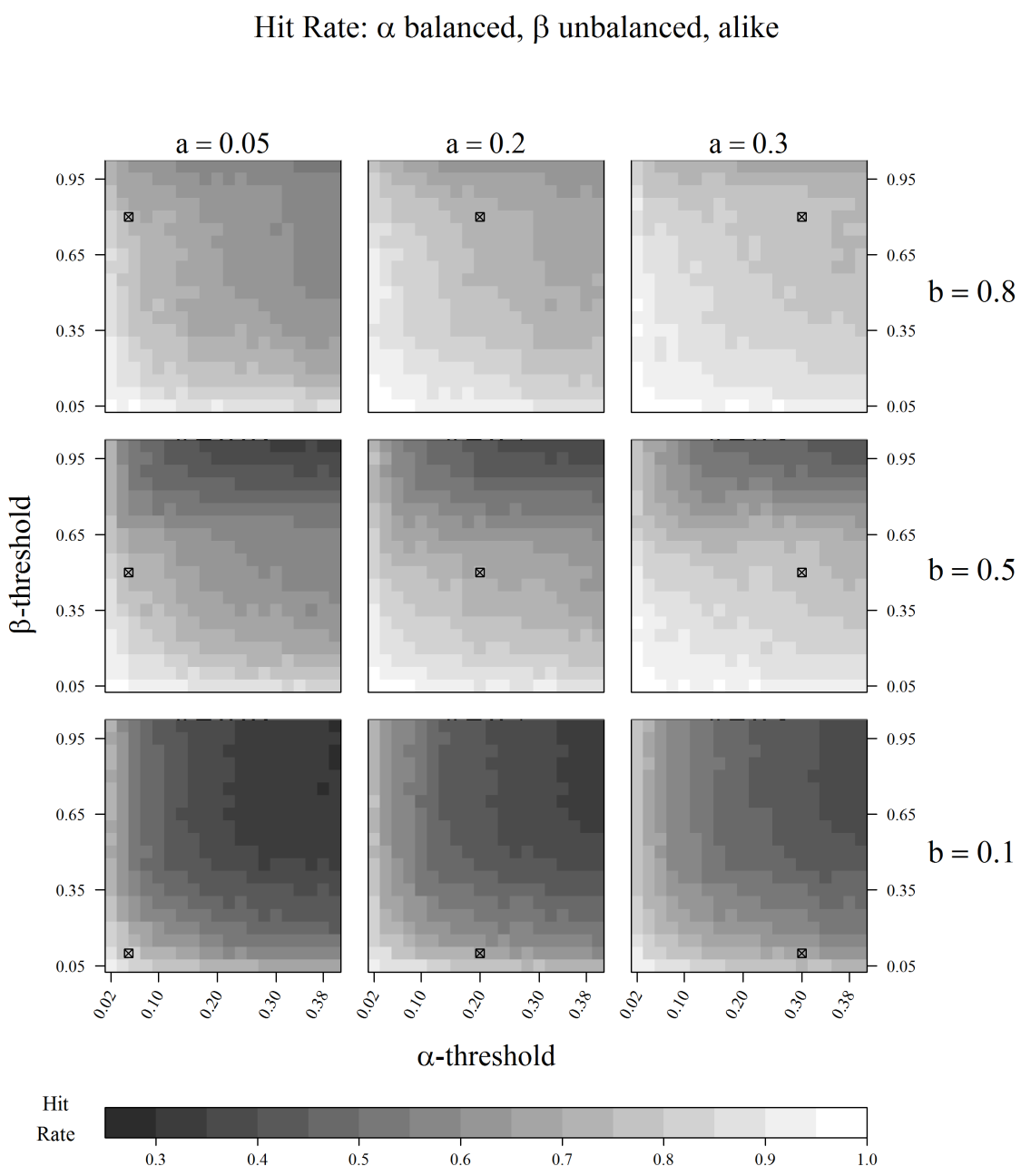

Figure S11: Hit rate for the best cluster in the condition of balanced α-DIF, unbalanced β-DIF, and same direction of DIF.

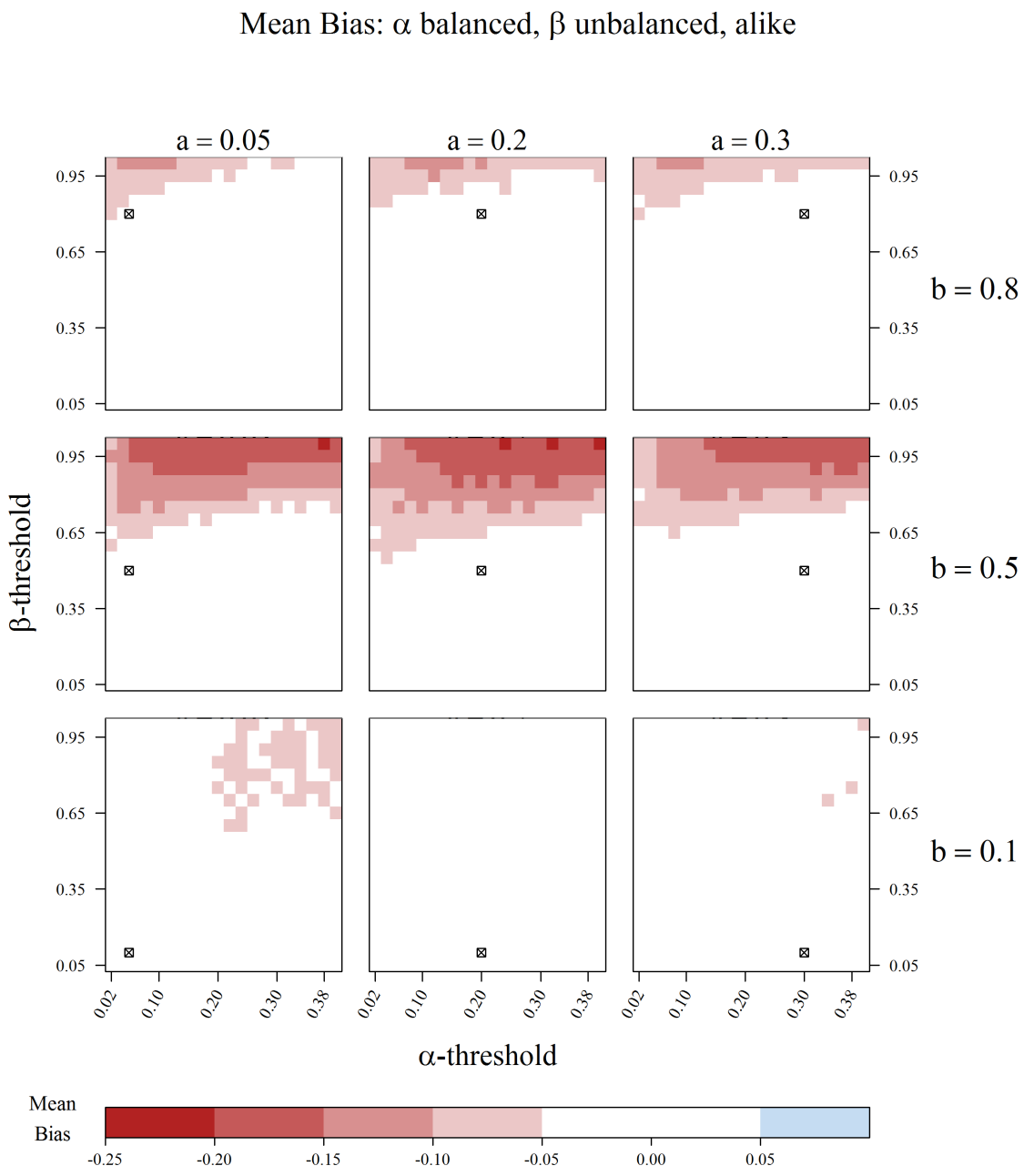

Figure S12: Bias in estimated mean difference for the best cluster in the condition of balanced α-DIF, unbalanced β-DIF, and same direction of DIF.

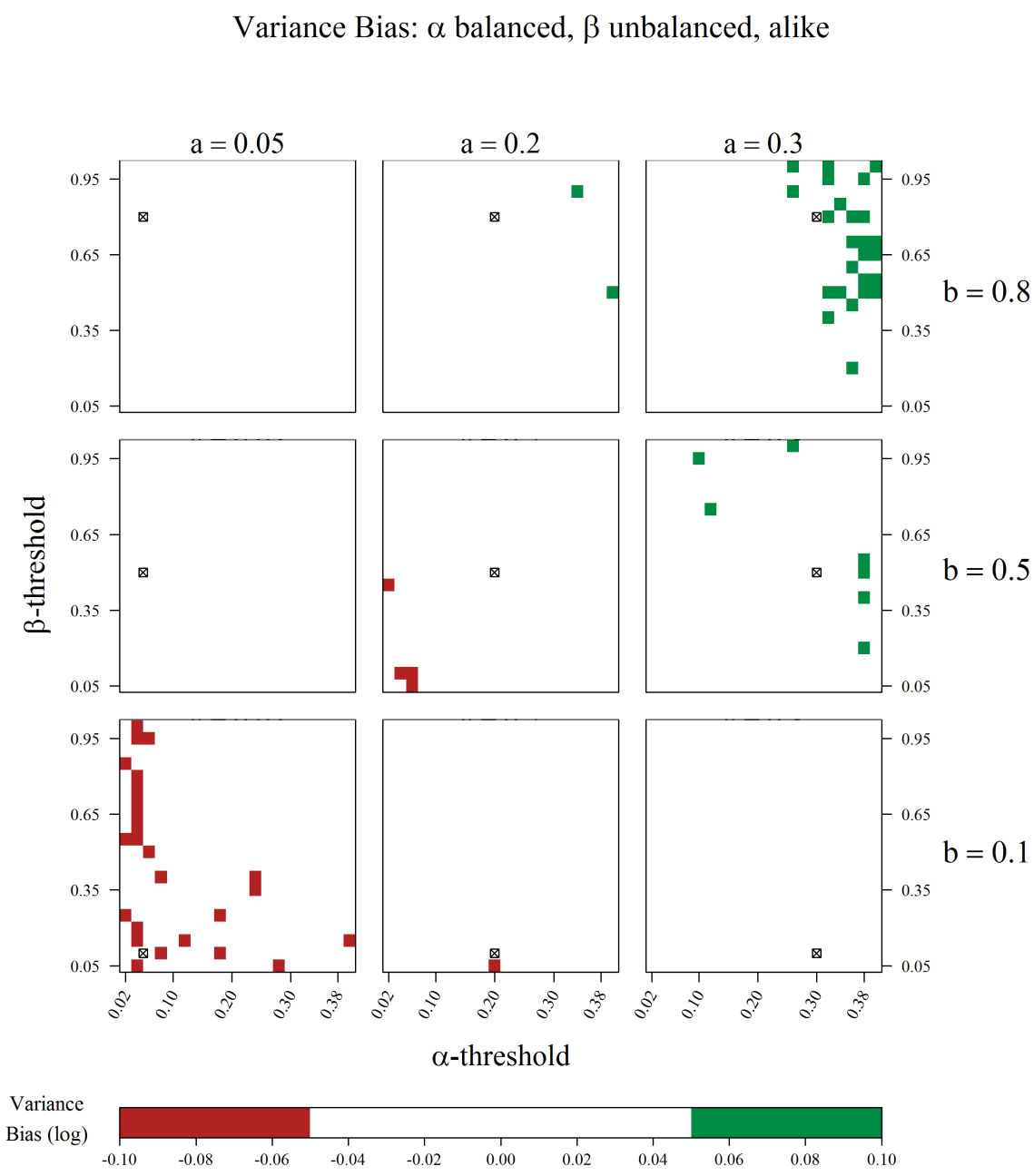

Figure S13: Bias in estimated relation of variances for the best cluster in the condition of balanced α-DIF, unbalanced β-DIF, and same direction of DIF.

α balanced, β unbalanced, opposed

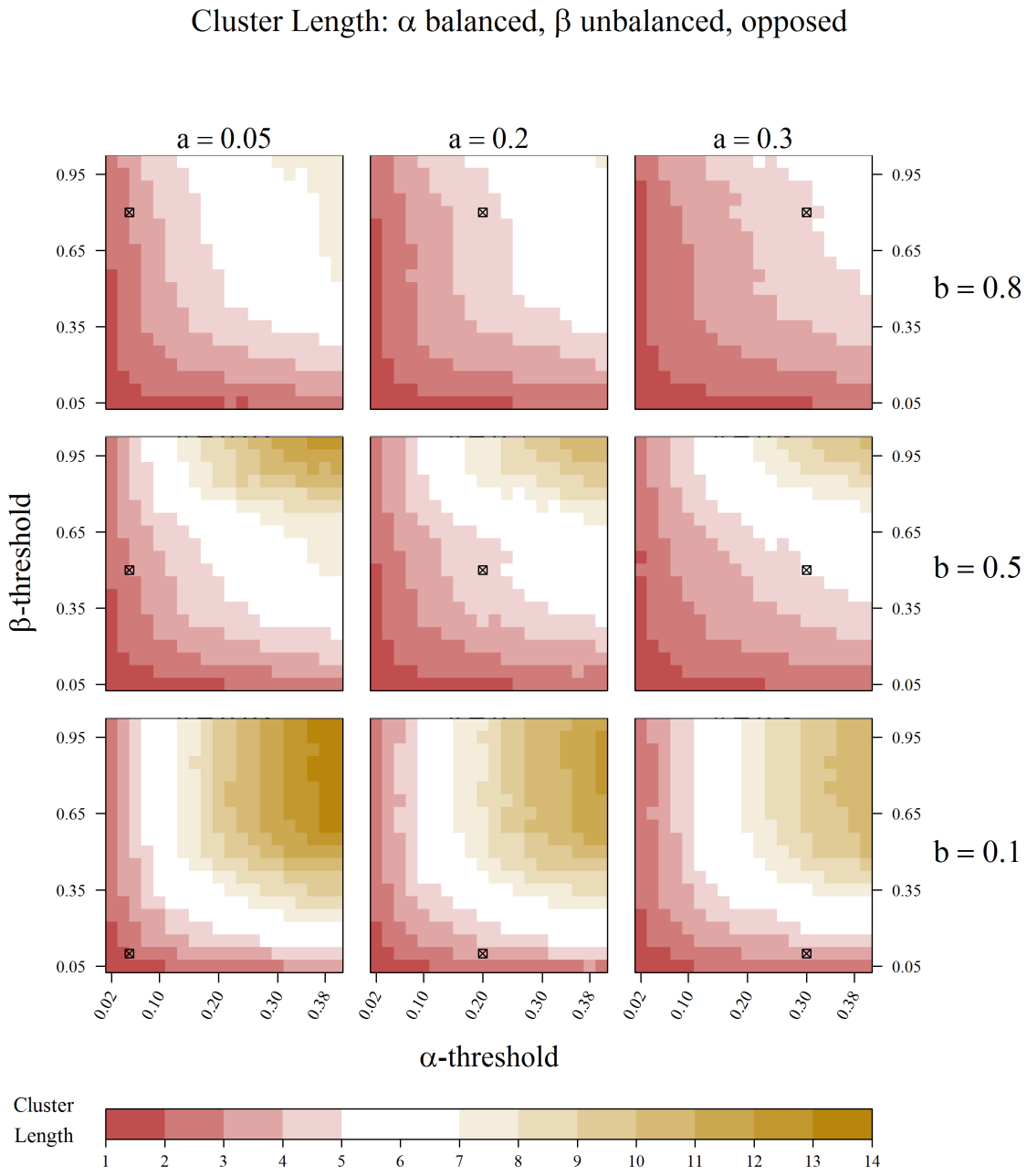

Figure S14: Cluster length for the best cluster in the condition of balanced α-DIF, unbalanced β-DIF, and opposed direction of DIF.

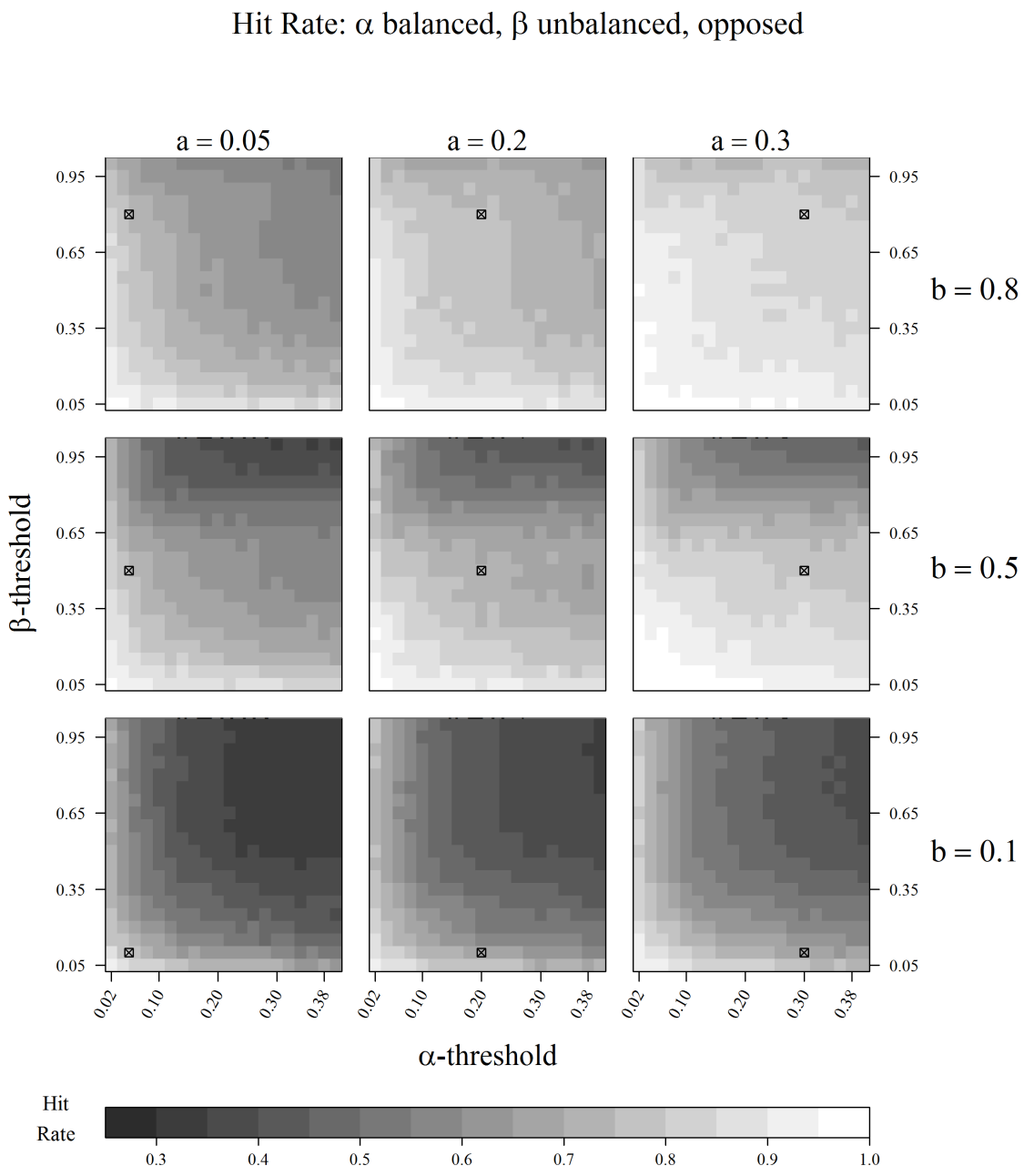

Figure S15: Hit rate for the best cluster in the condition of balanced α-DIF, unbalanced β-DIF, and opposed direction of DIF.

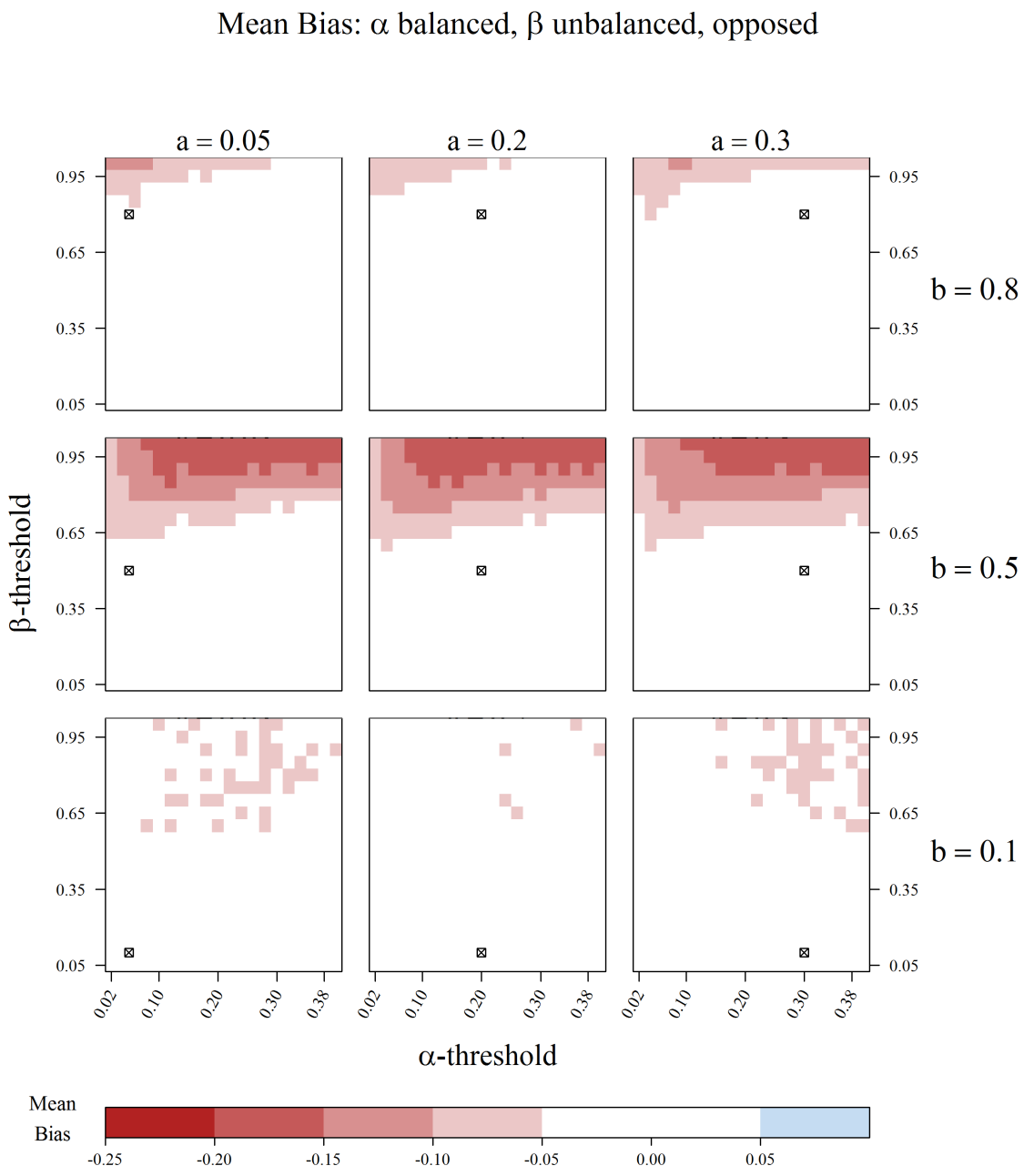

Figure S16: Bias in estimated mean difference for the best cluster in the condition of balanced α-DIF, unbalanced β-DIF, and opposed direction of DIF.

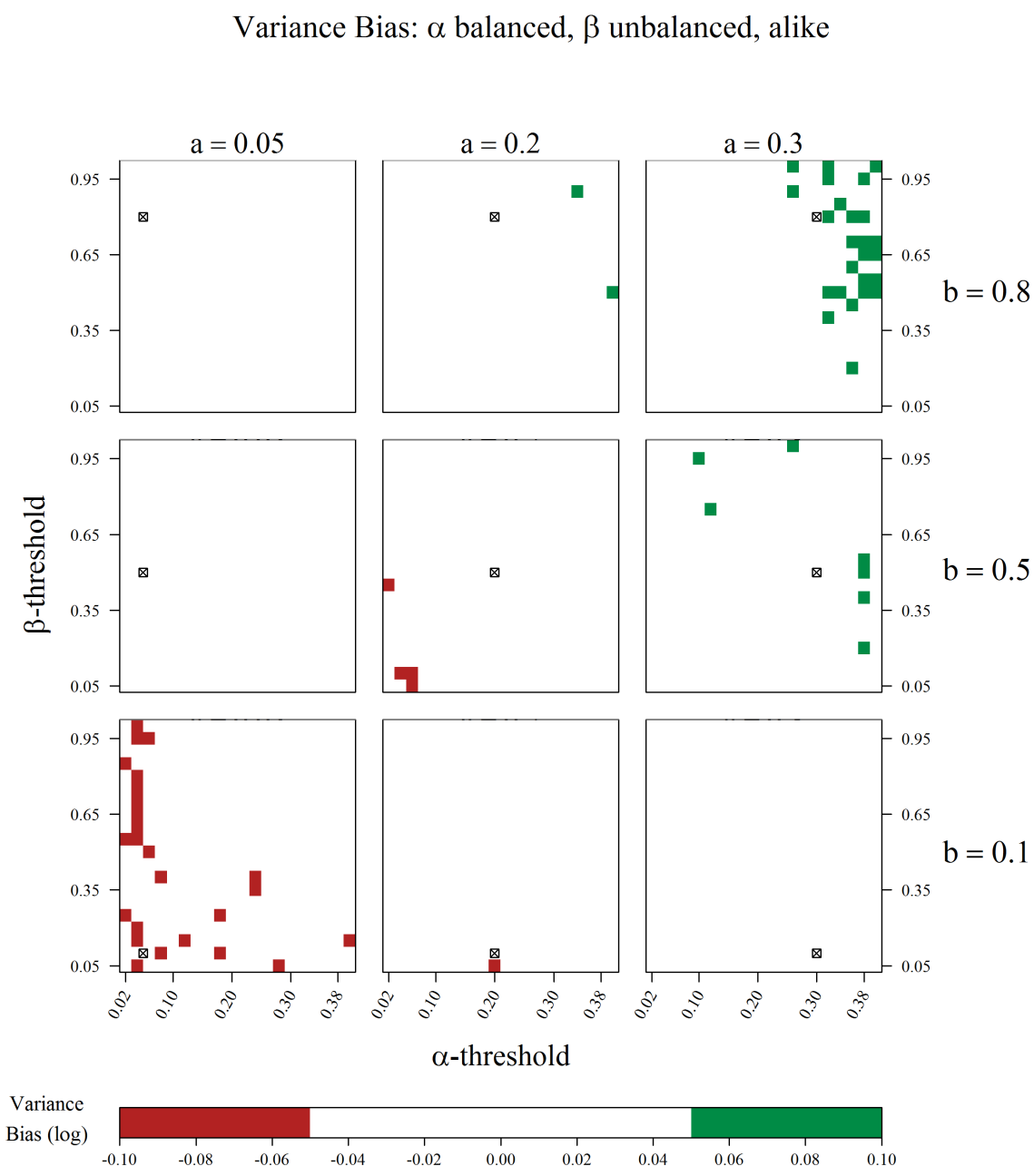

Figure S17: Bias in estimated relation of variances for the best cluster in the condition of balanced α-DIF, unbalanced β-DIF, and opposed direction of DIF.

α unbalanced, β balanced, alike

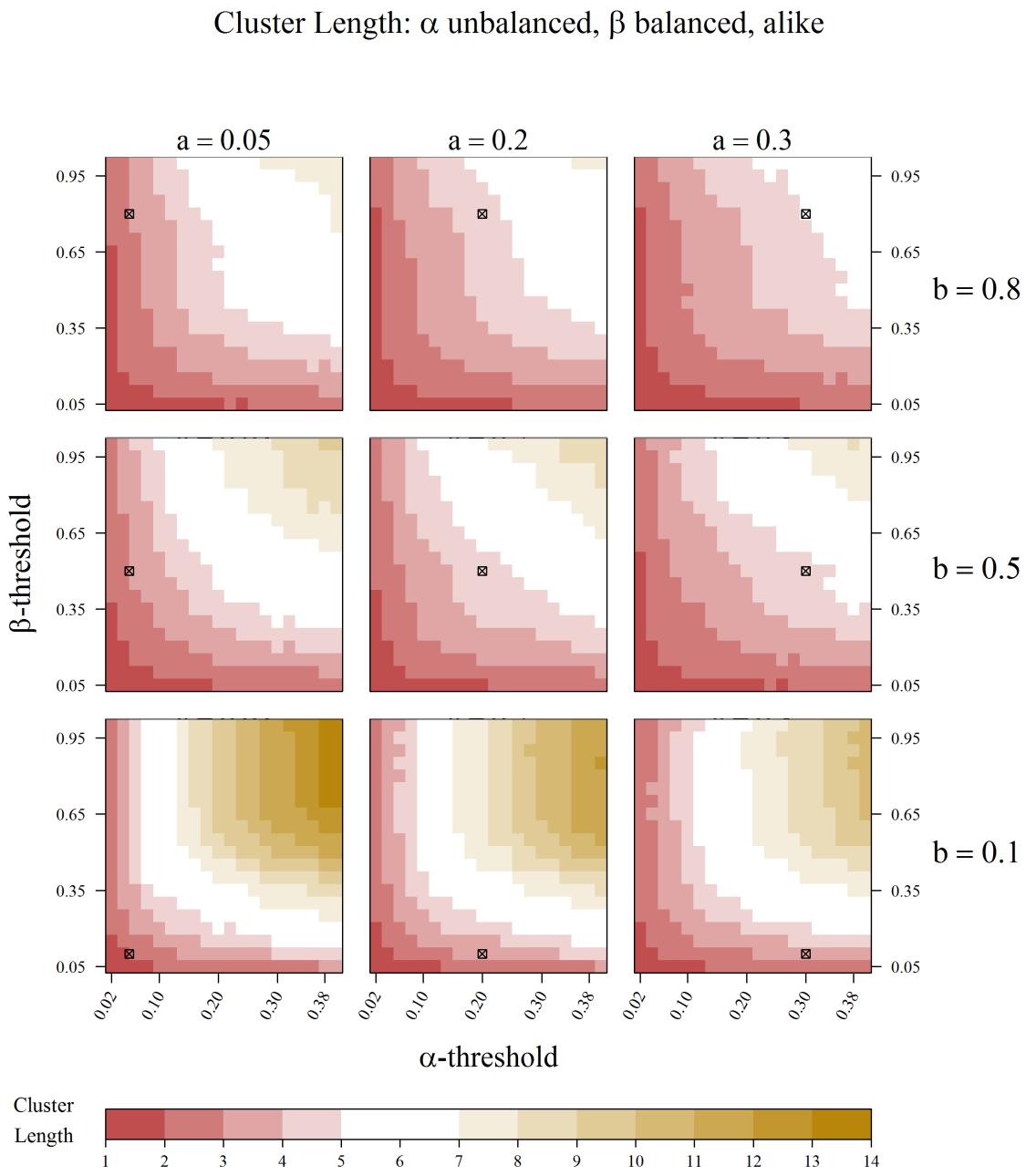

Figure S18: Cluster length for the best cluster in the condition of unbalanced α-DIF, balanced β-DIF, and same direction of DIF.

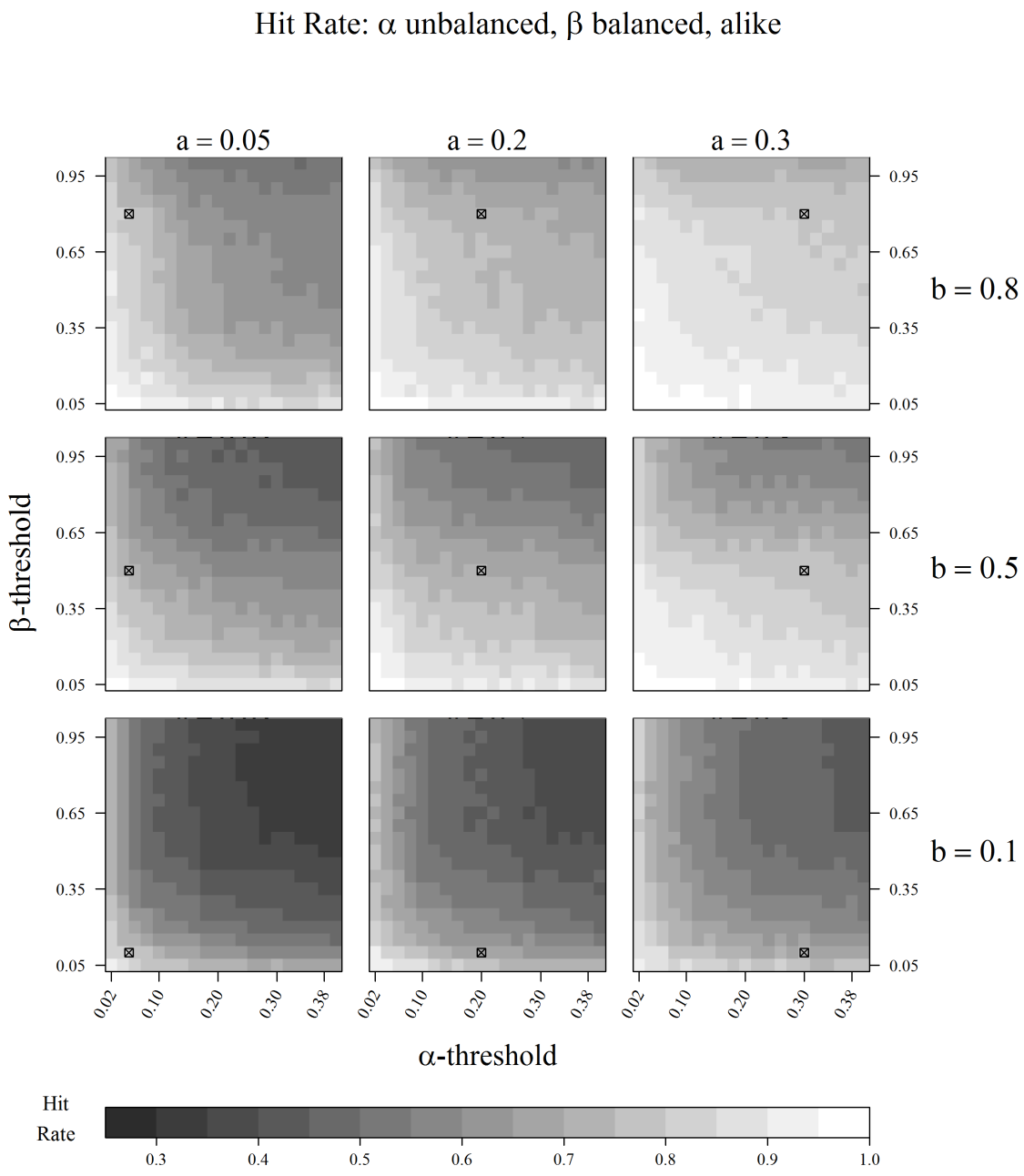

Figure S19: Hit rate for the best cluster in the condition of unbalanced α-DIF, balanced β-DIF, and same direction of DIF.

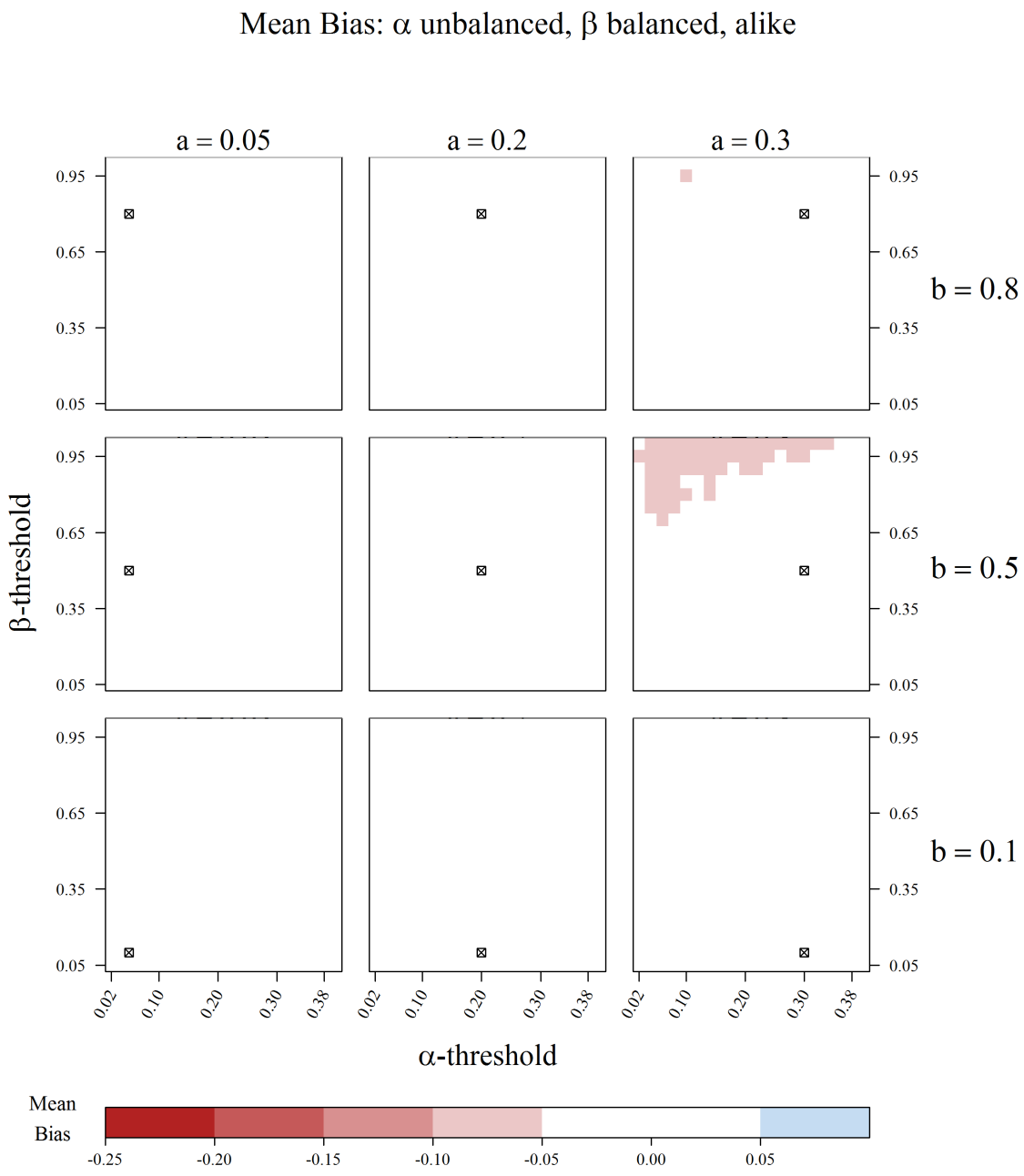

Figure S20: Bias in estimated mean difference for the best cluster in the condition of unbalanced α-DIF, balanced β-DIF, and same direction of DIF.

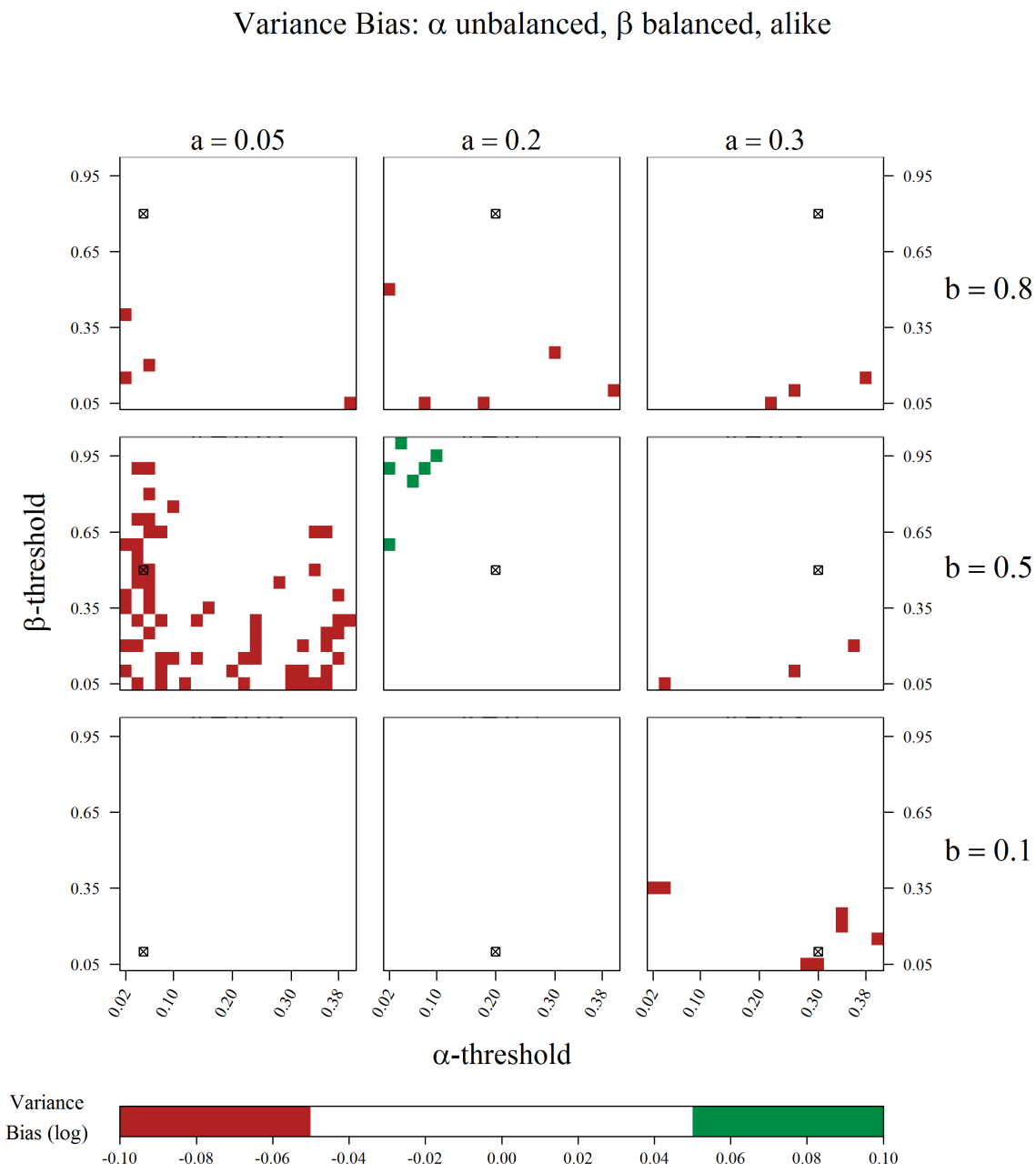

Figure S21: Bias in estimated relation of variances for the best cluster in the condition of unbalanced α-DIF, balanced β-DIF, and same direction of DIF.

α unbalanced, β balanced, opposed

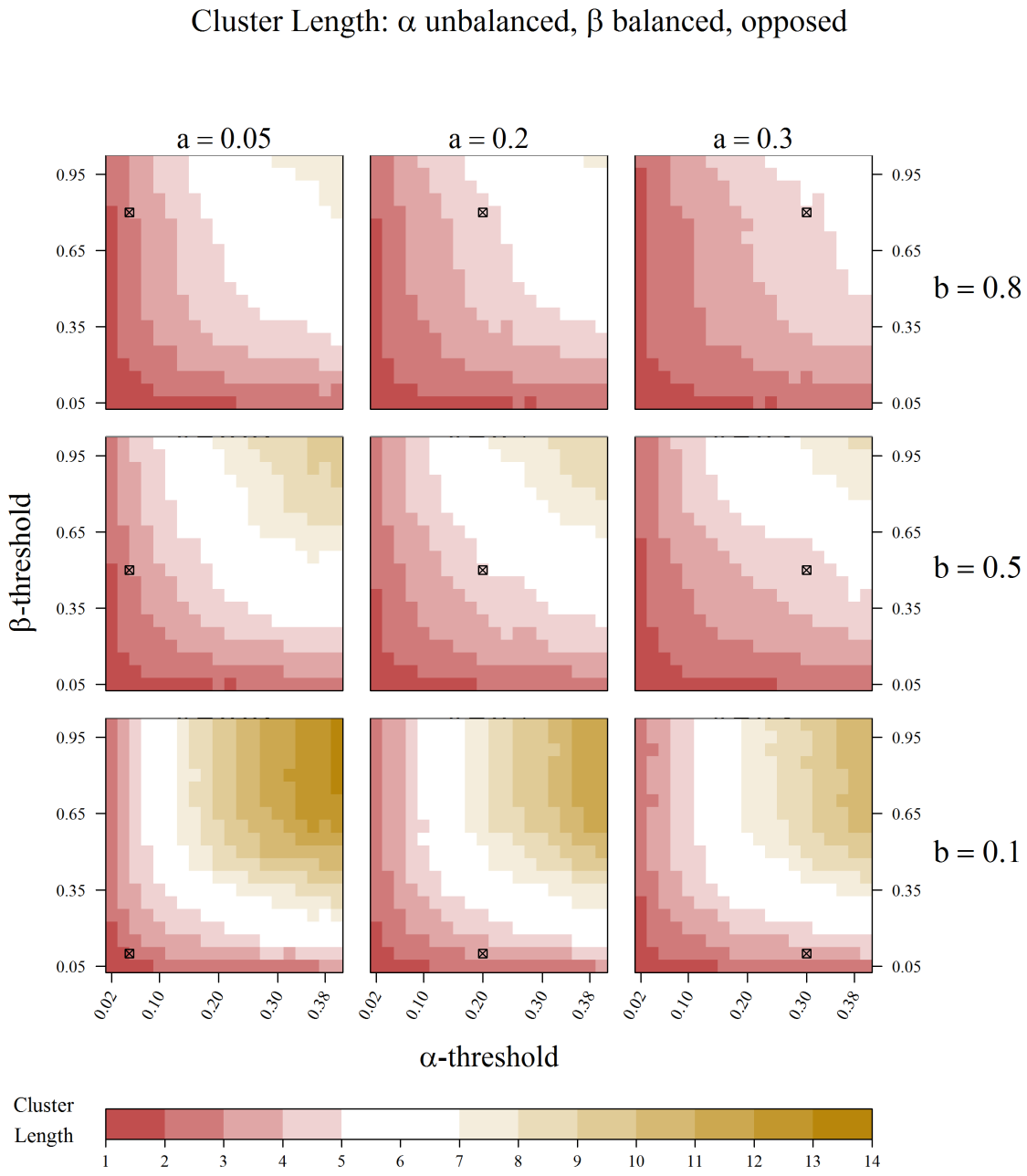

Figure S22: Cluster length for the best cluster in the condition of unbalanced α-DIF, balanced β-DIF, and opposed direction of DIF.

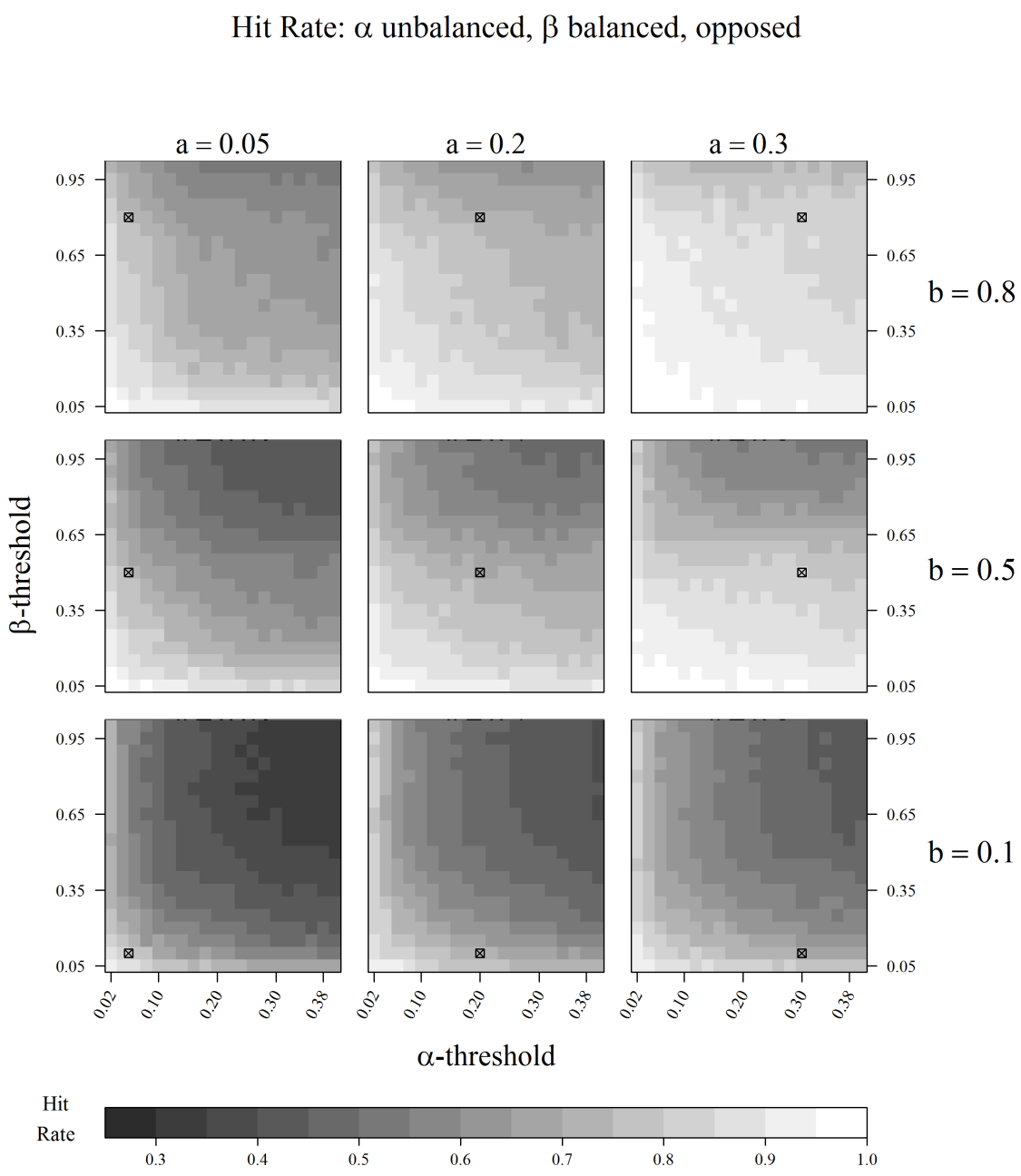

Figure S23: Hit rate for the best cluster in the condition of unbalanced α-DIF, balanced β-DIF, and opposed direction of DIF.

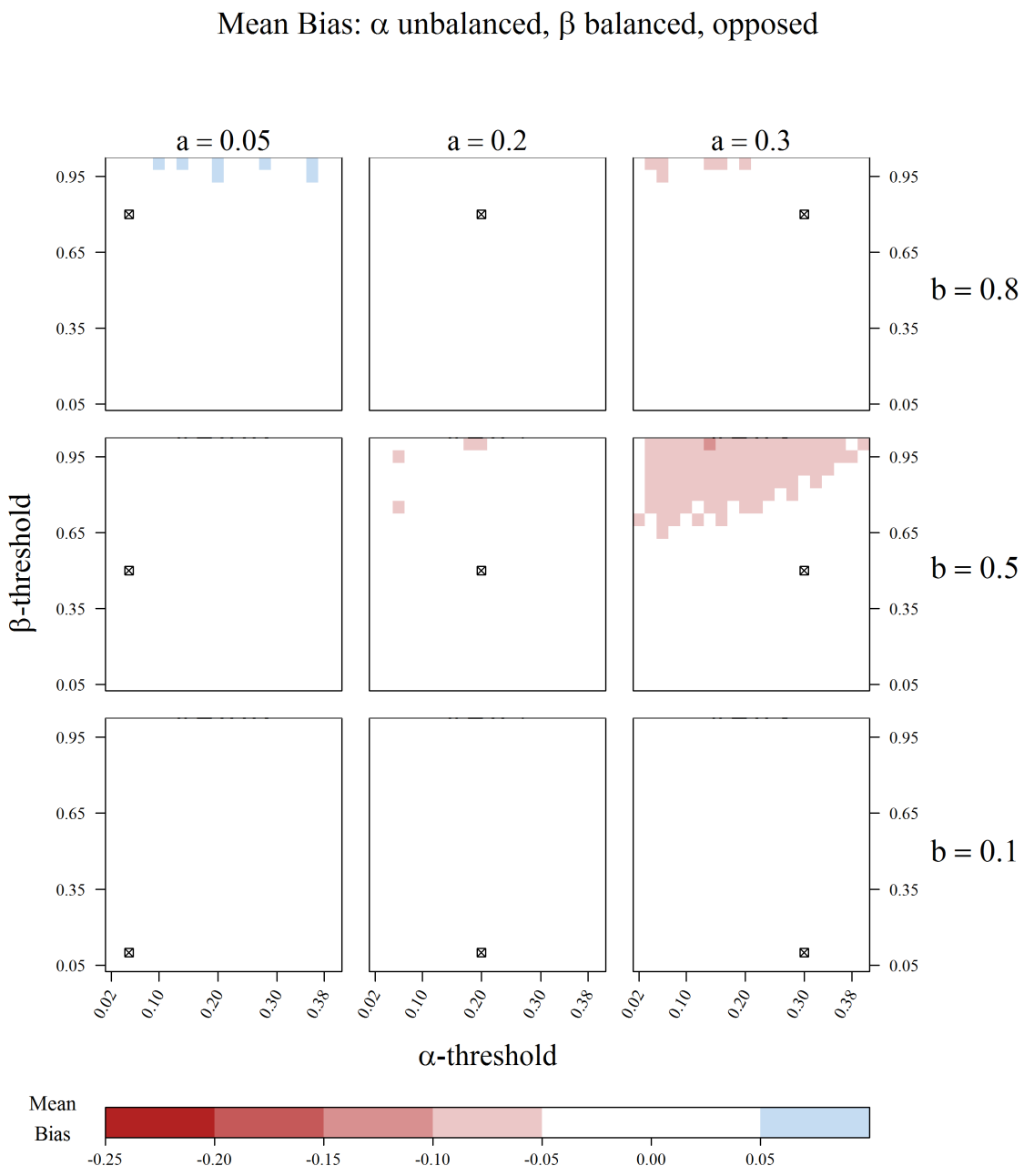

Figure S24: Bias in estimated mean difference for the best cluster in the condition of unbalanced α-DIF, balanced β-DIF, and opposed direction of DIF.

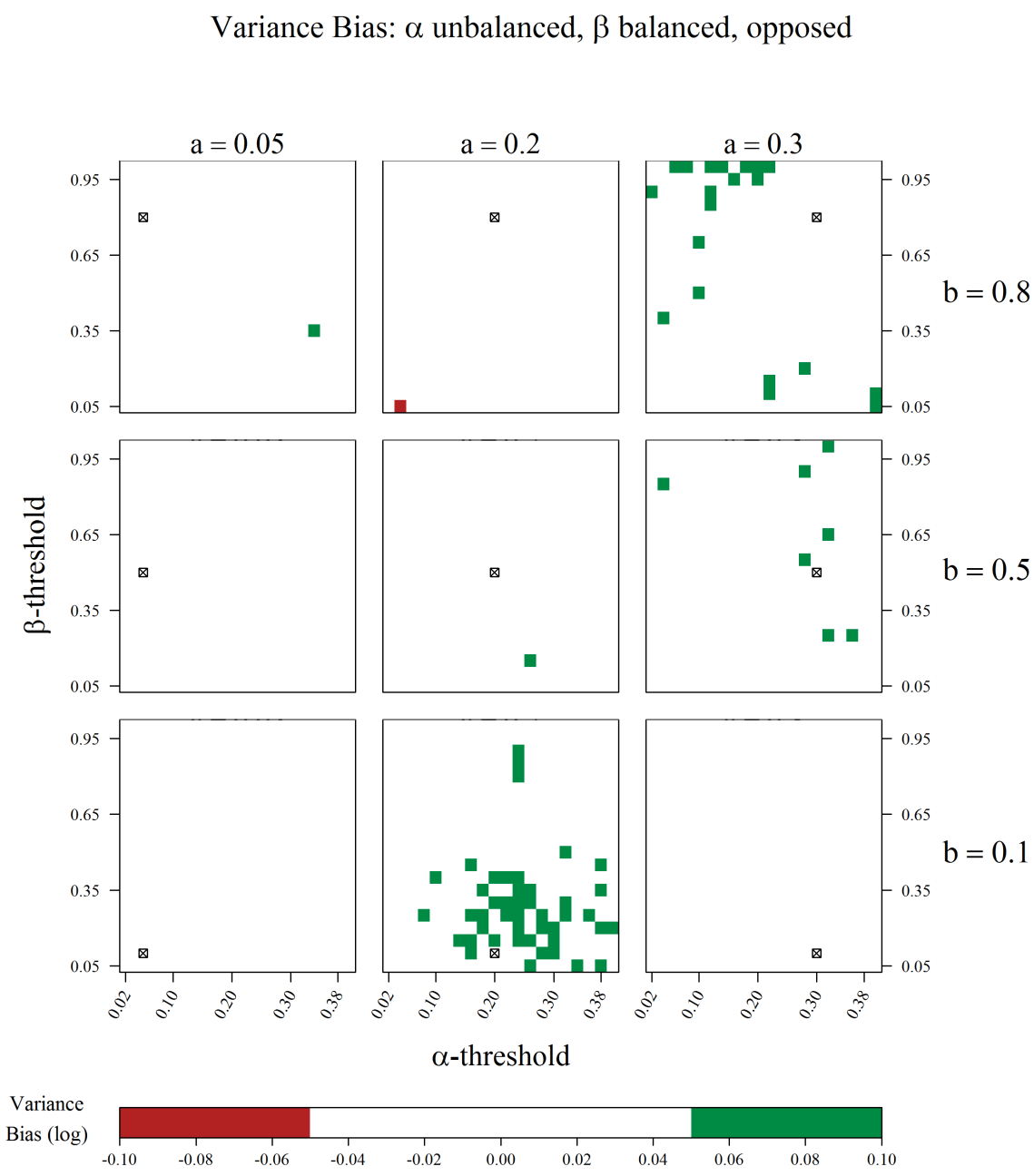

Figure S25: Bias in estimated relation of variances for the best cluster in the condition of unbalanced α-DIF, balanced β-DIF, and opposed direction of DIF.

α unbalanced, β unbalanced, alike

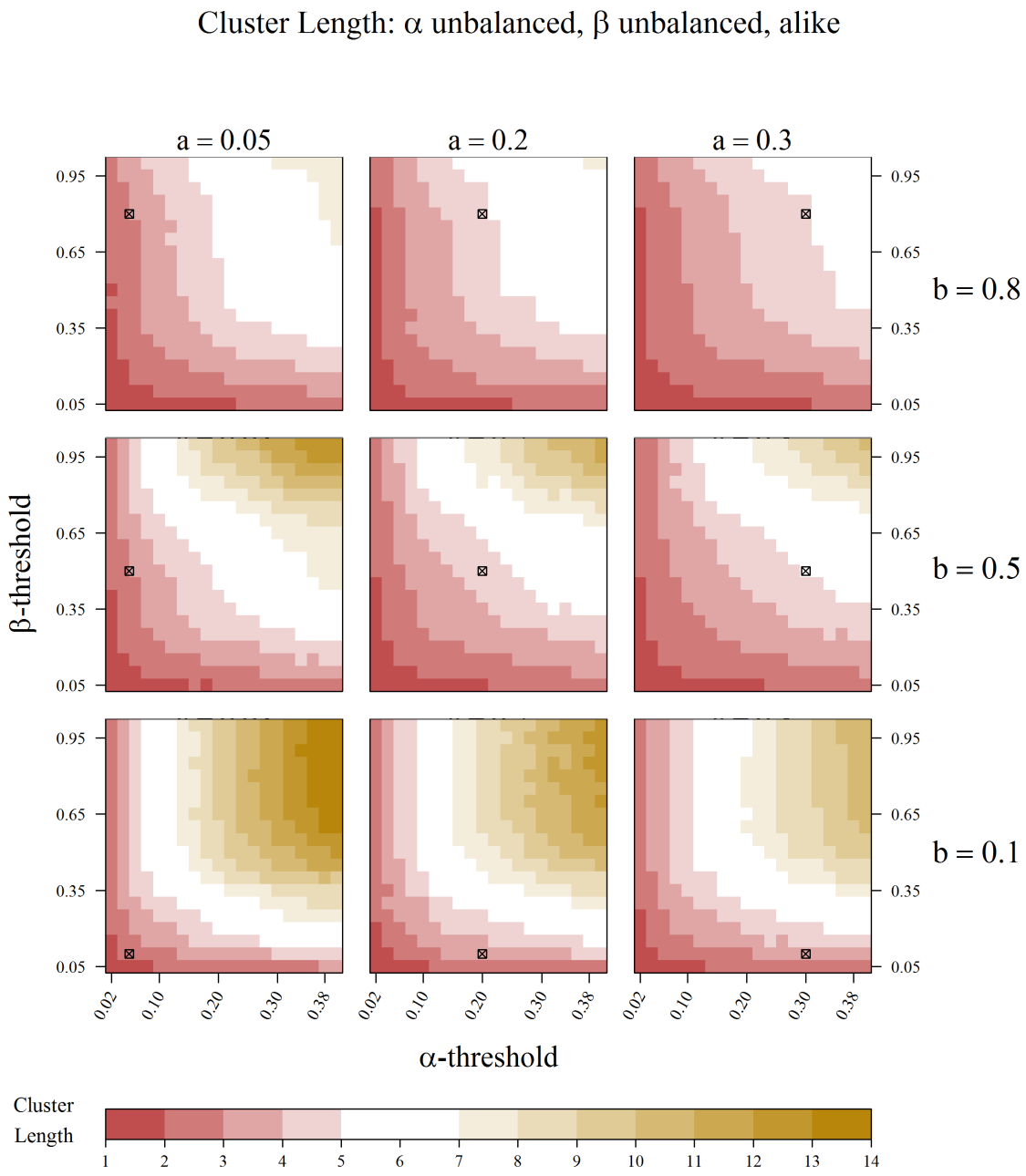

Figure S26: Cluster length for the best cluster in the condition of unbalanced α-DIF, unbalanced β-DIF, and same direction of DIF.

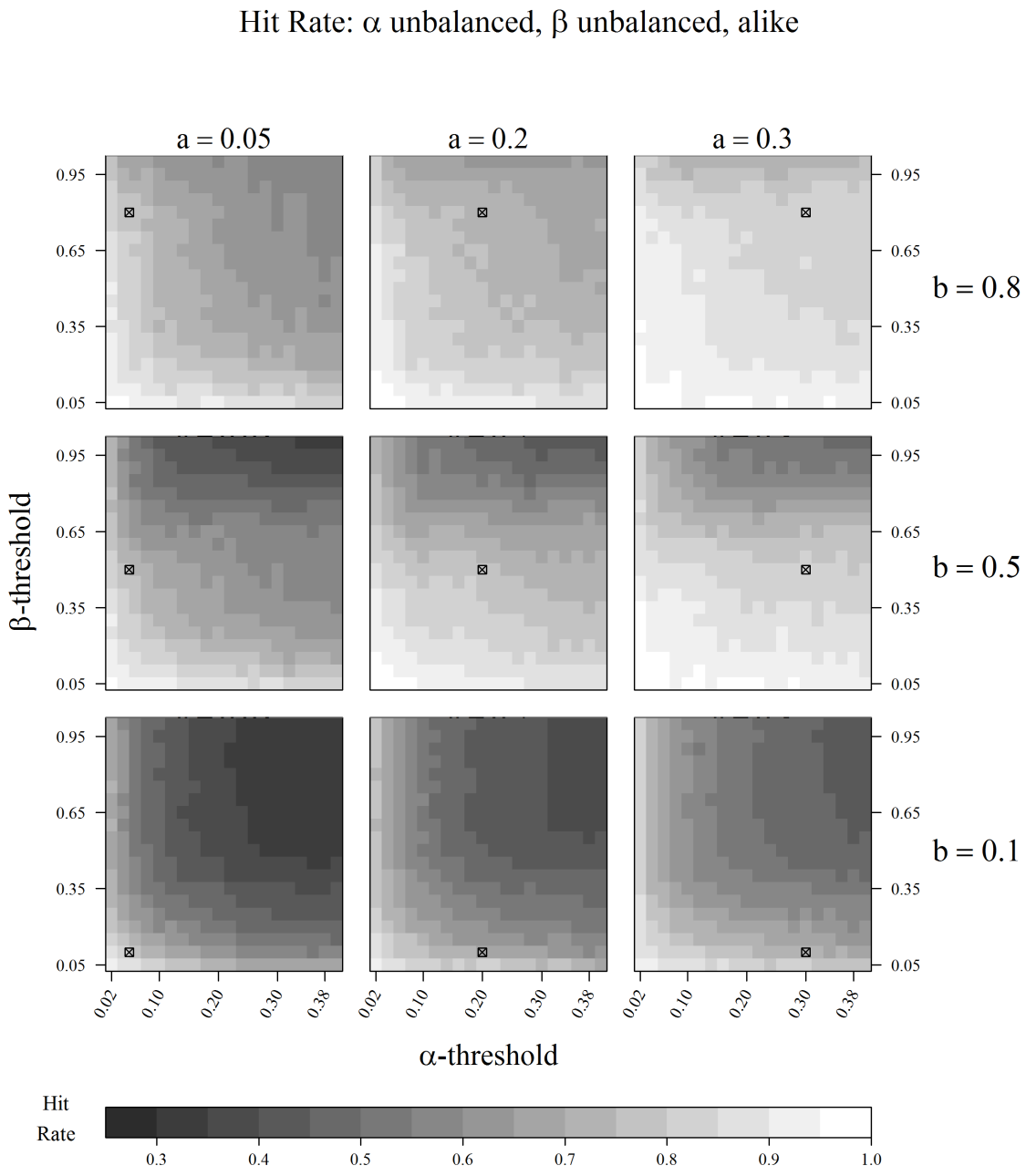

Figure S27: Hit rate for the best cluster in the condition of unbalanced α-DIF, unbalanced β-DIF, and same direction of DIF.

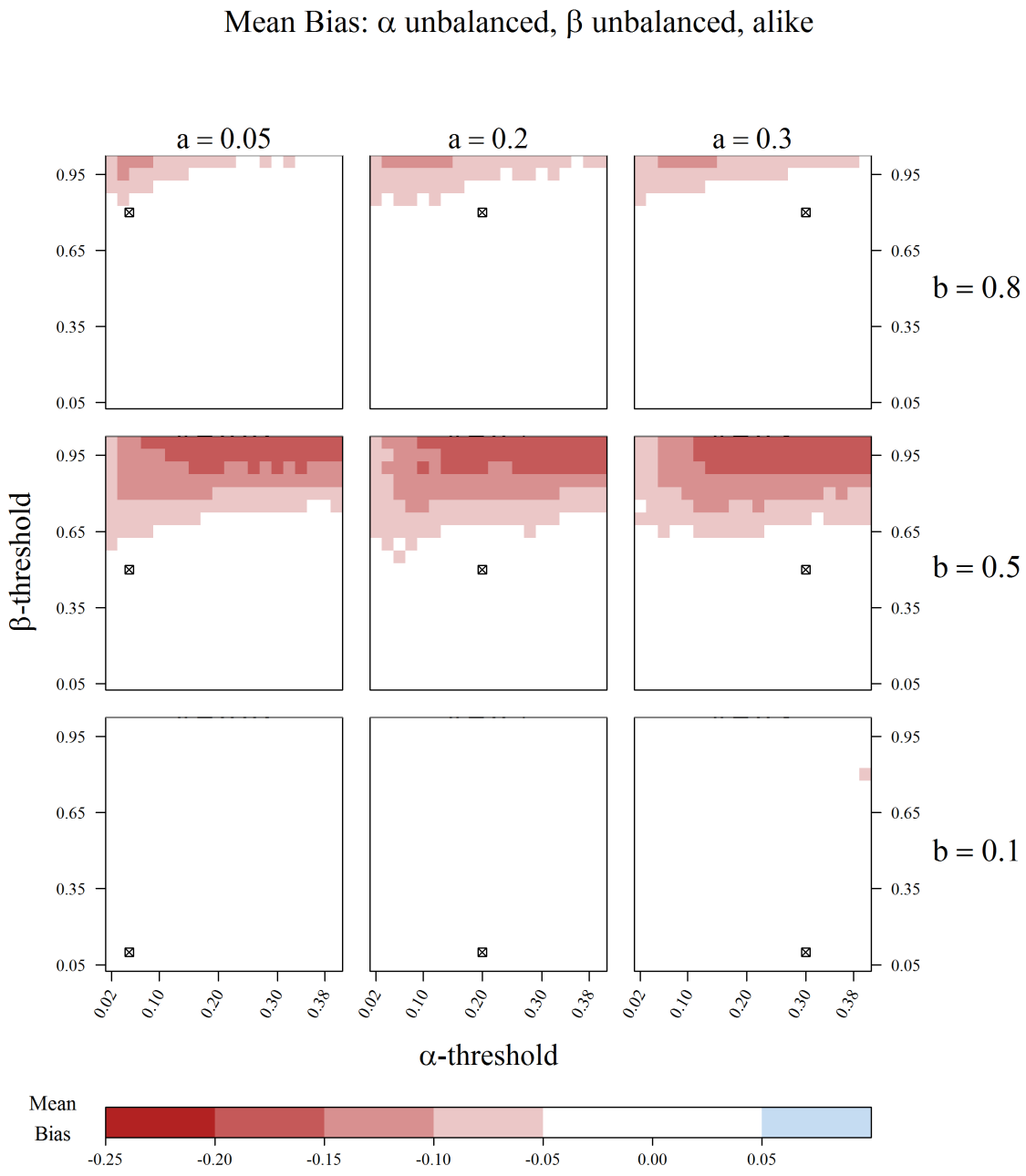

Figure S28: Bias in estimated mean difference for the best cluster in the condition of unbalanced α-DIF, unbalanced β-DIF, and same direction of DIF.

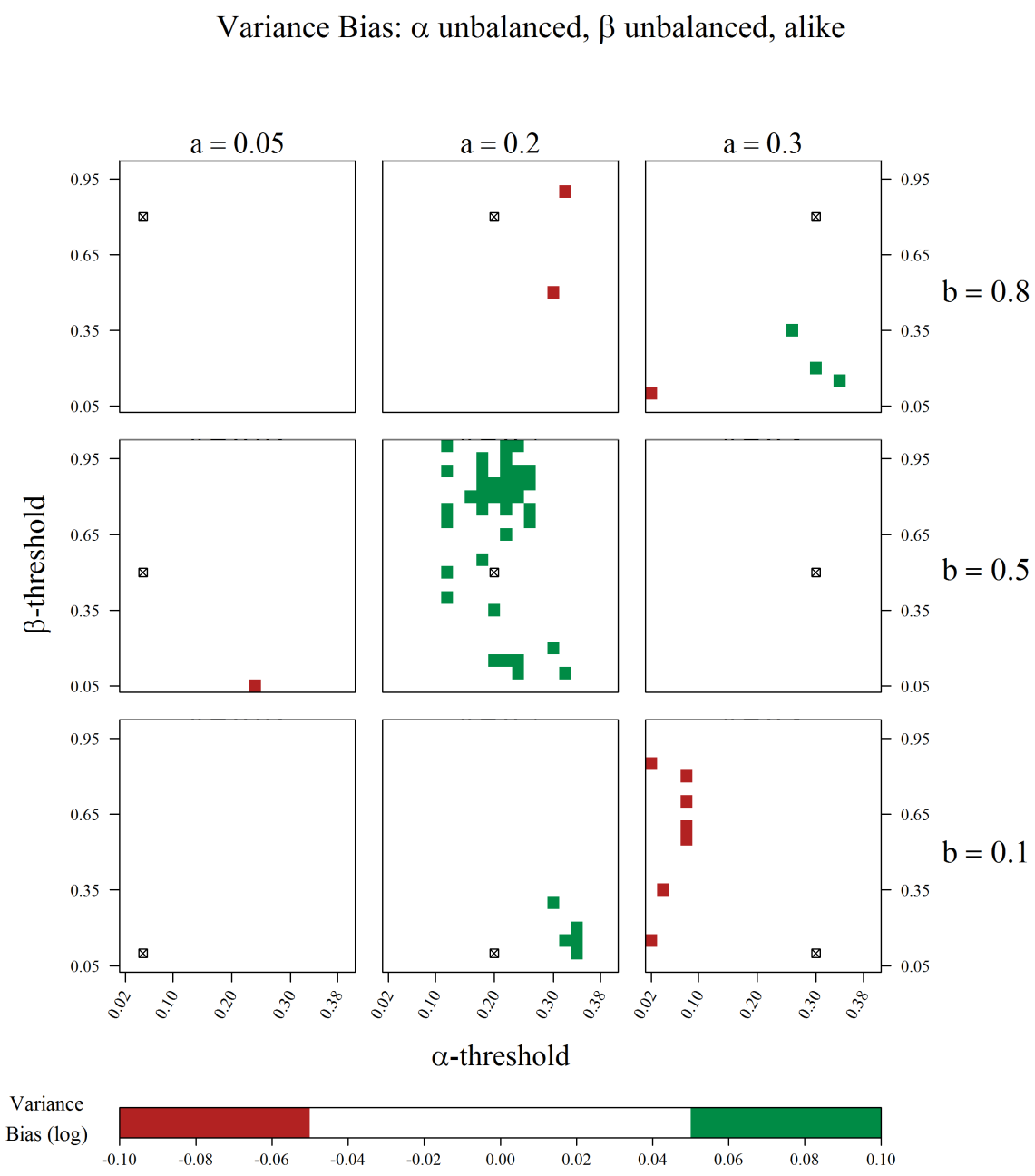

Figure S29: Bias in estimated relation of variances for the best cluster in the condition of unbalanced α-DIF, unbalanced β-DIF, and same direction of DIF.

α unbalanced, β unbalanced, opposed

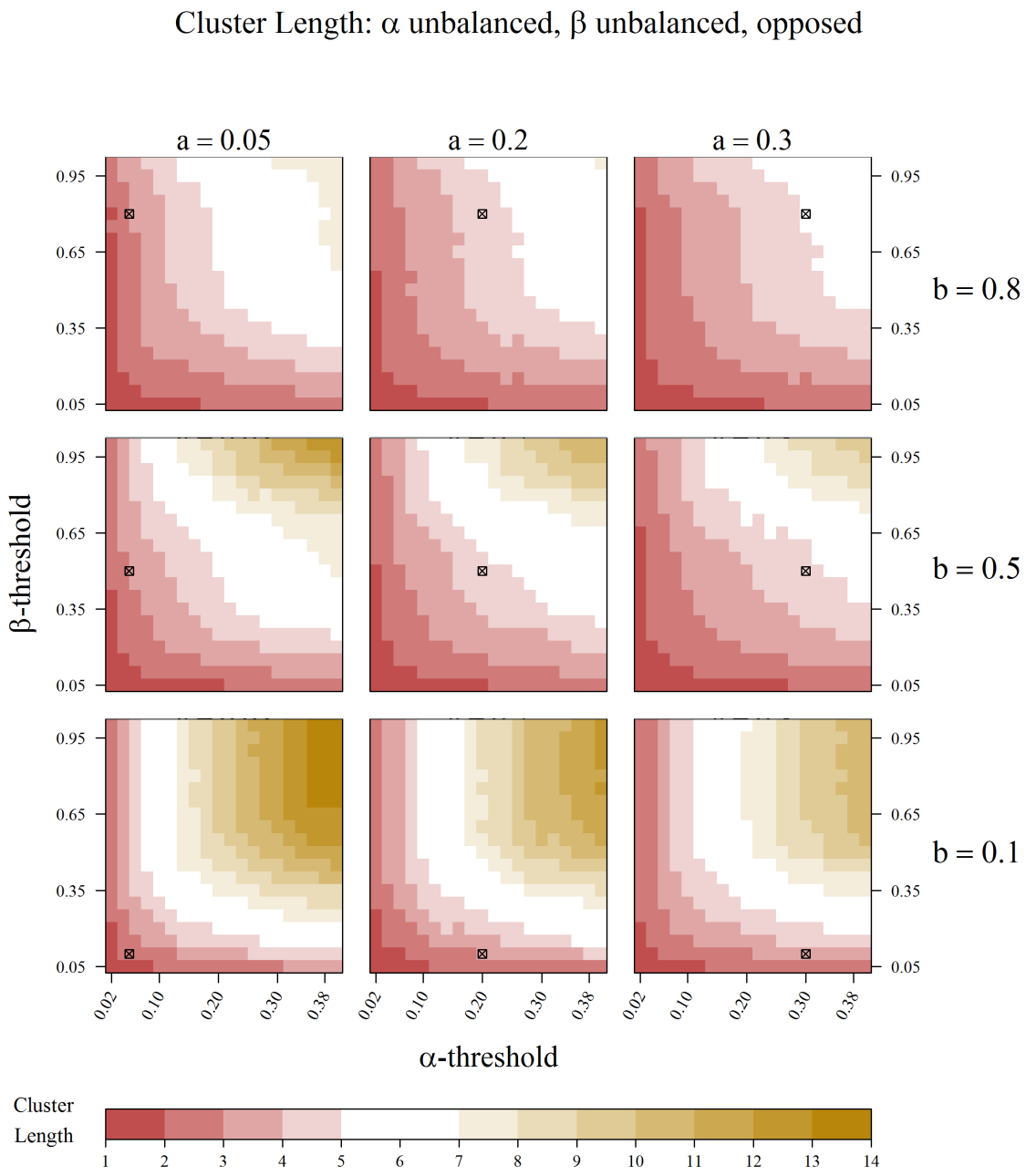

Figure S30: Cluster length for the best cluster in the condition of unbalanced α-DIF, unbalanced β-DIF, and opposed direction of DIF.

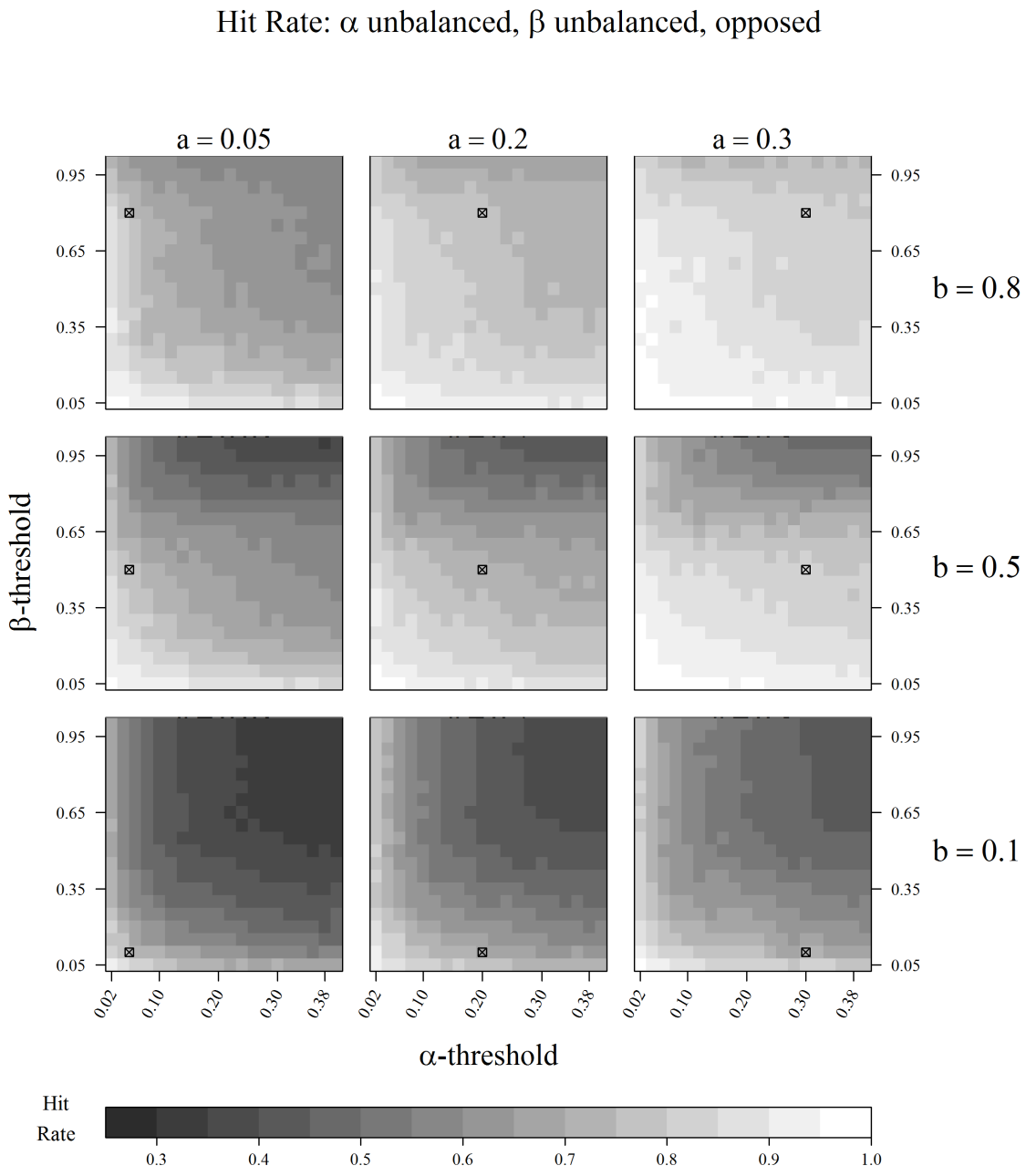

Figure S31: Hit rate for the best cluster in the condition of unbalanced α-DIF, unbalanced β-DIF, and opposed direction of DIF.

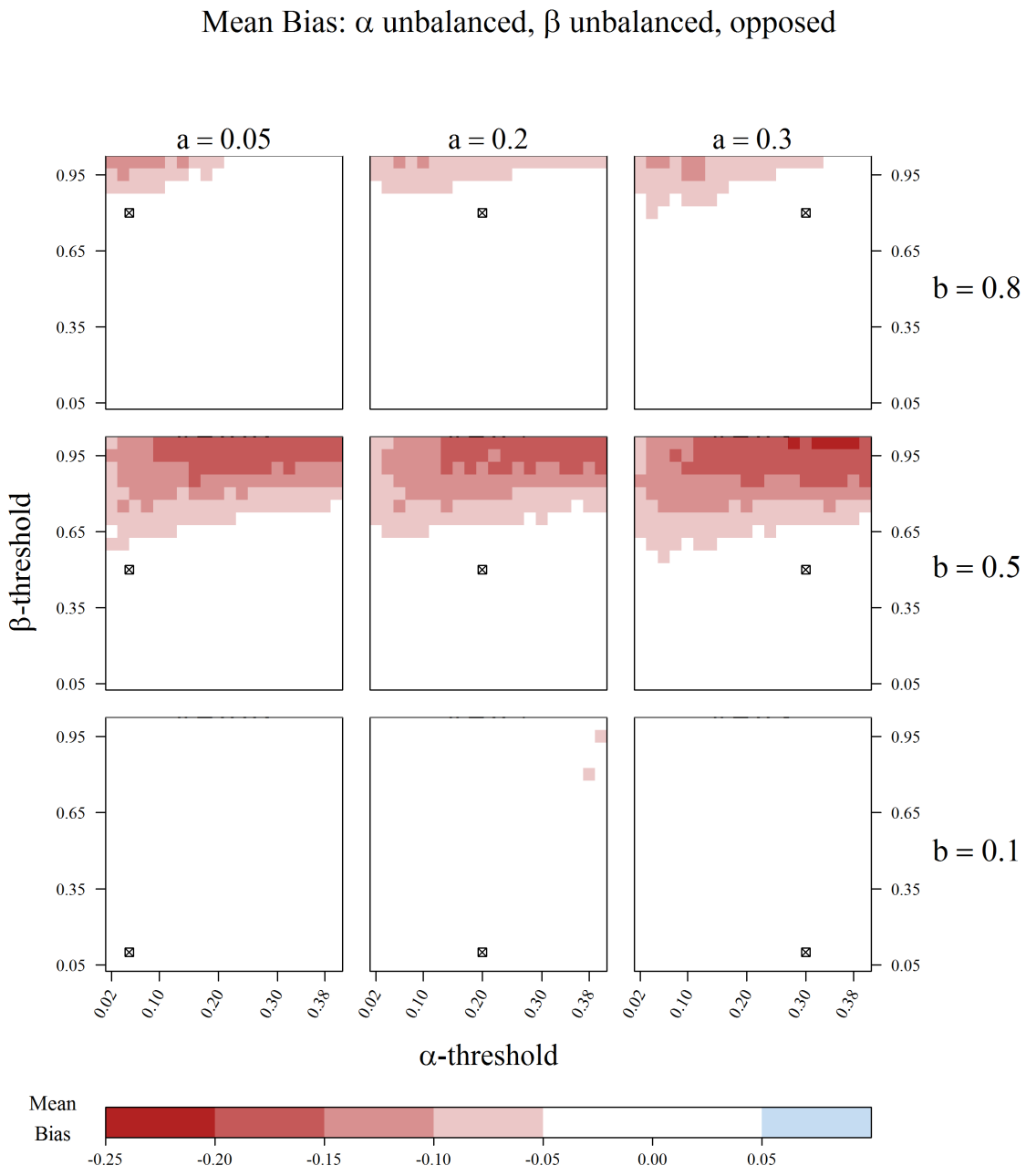

Figure S32: Bias in estimated mean difference for the best cluster in the condition of unbalanced α-DIF, unbalanced β-DIF, and opposed direction of DIF.

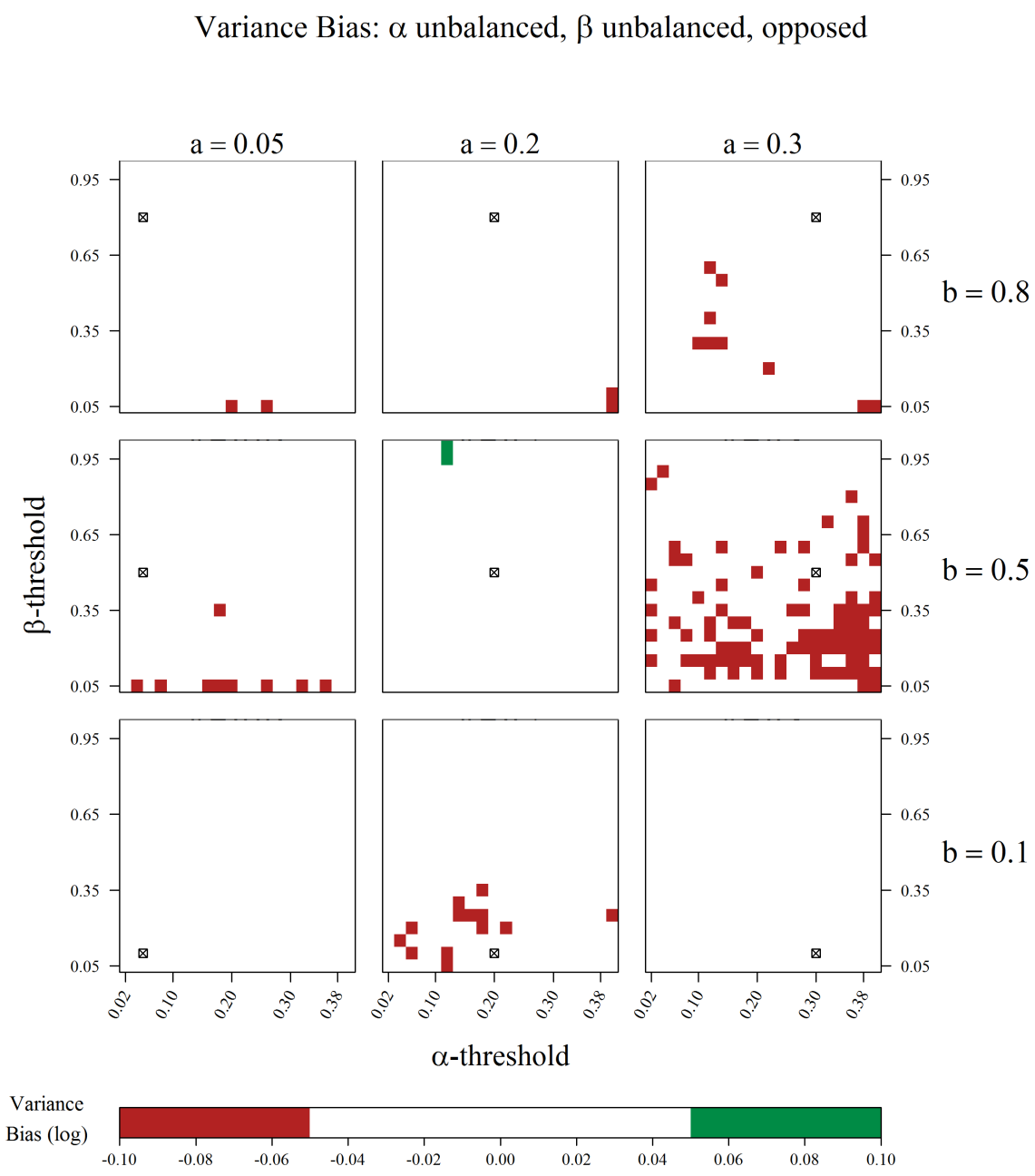

Figure S33: Bias in estimated relation of variances for the best cluster in the condition of unbalanced α-DIF, unbalanced β-DIF, and opposed direction of DIF.

Results compared across DIF-balancedness conditions

a=0.05, b=0.10

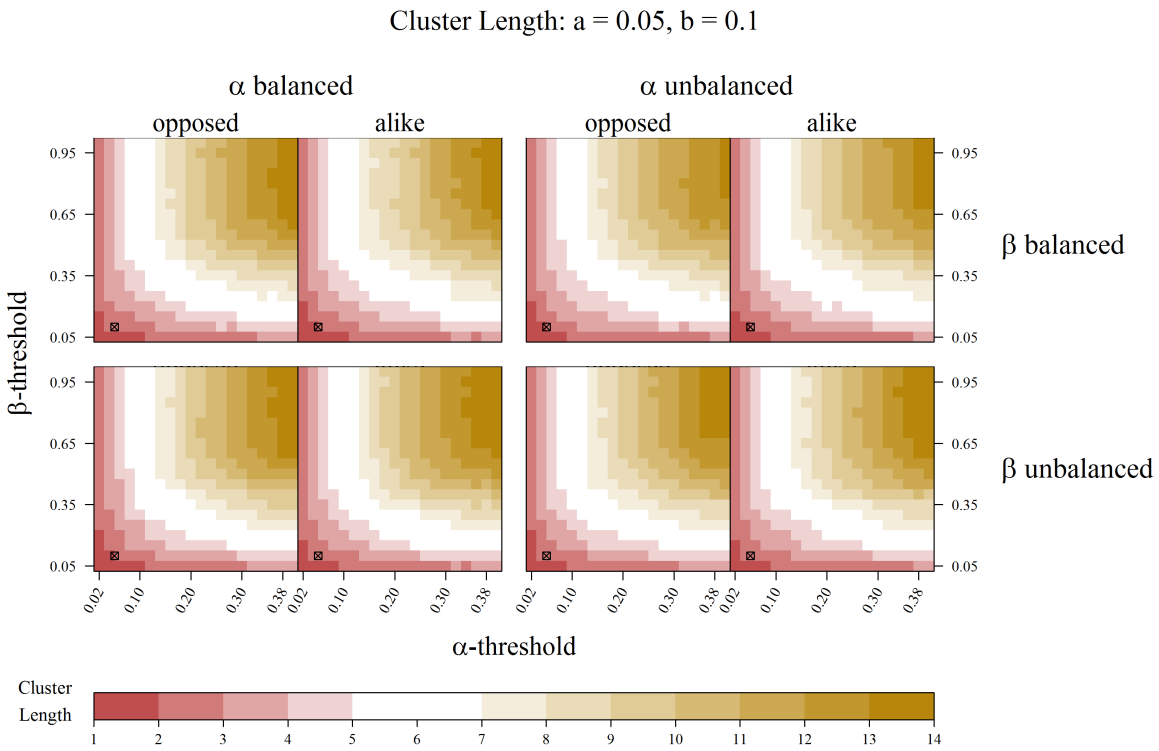

Figure S34: Cluster length for the best cluster for $a = 0.05, b = 0.10$.

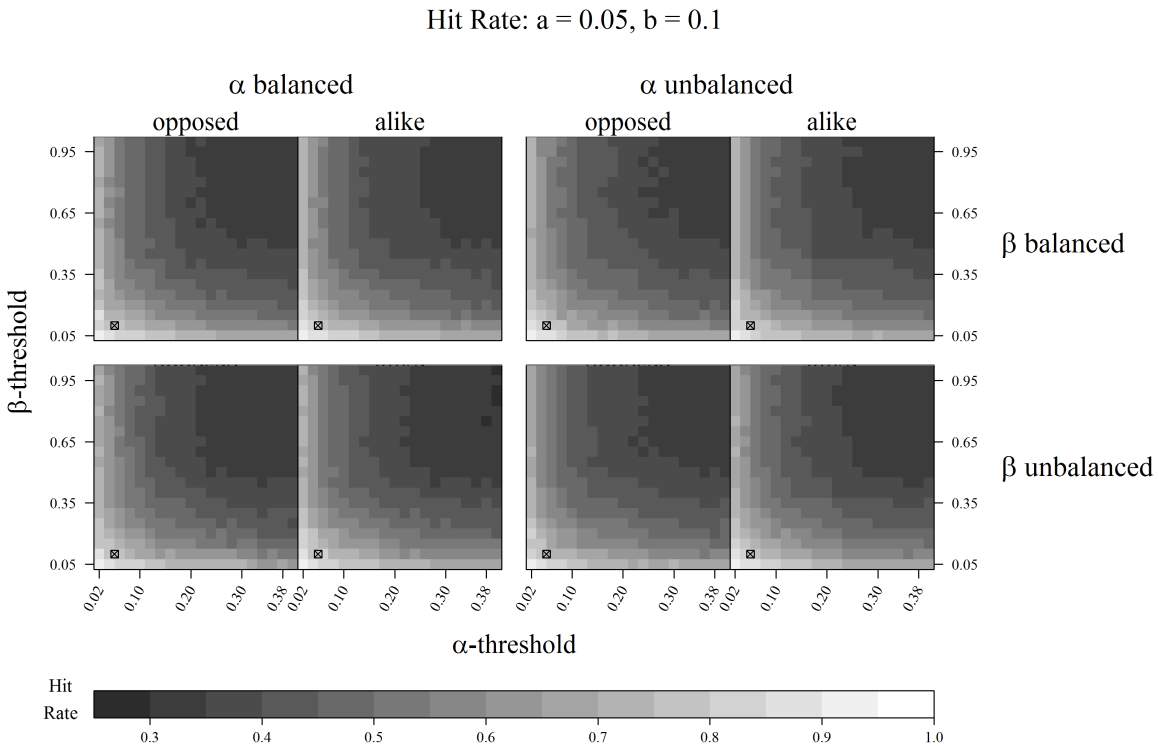

Figure S35: Hit rate for the best cluster for $a = 0.05, b = 0.10$.

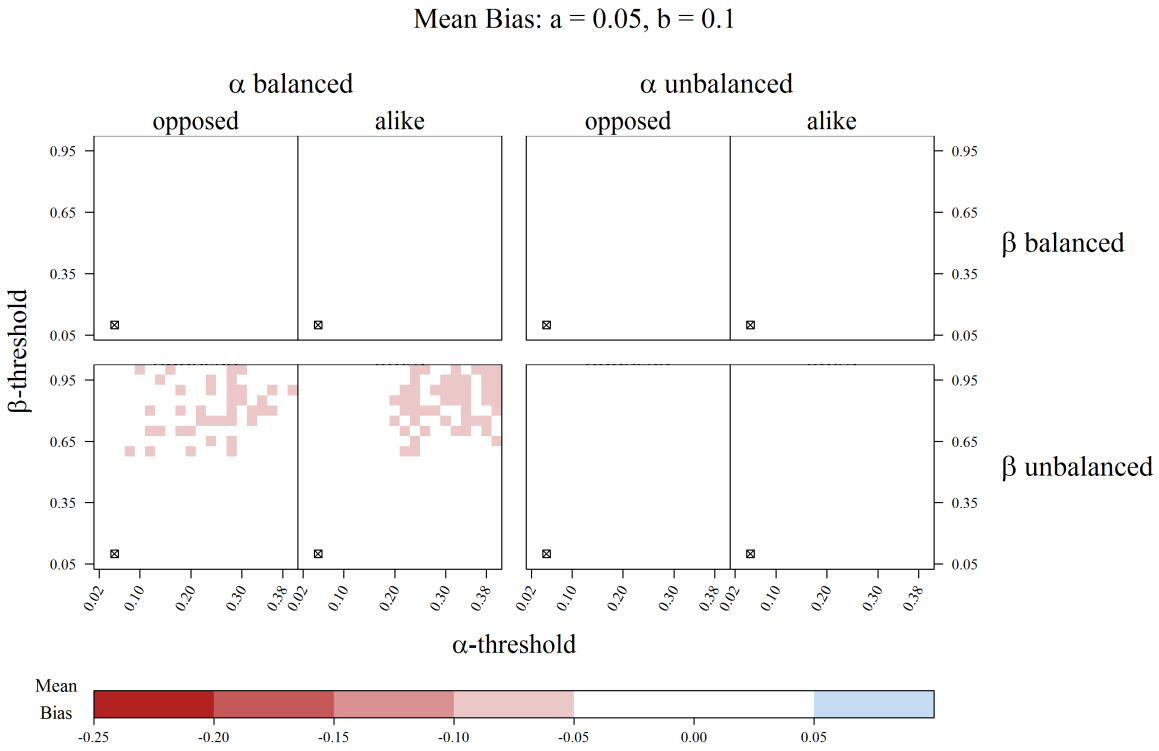

Figure S36: Bias in estimated mean difference for the best cluster for $a = 0.05, b = 0.10$.

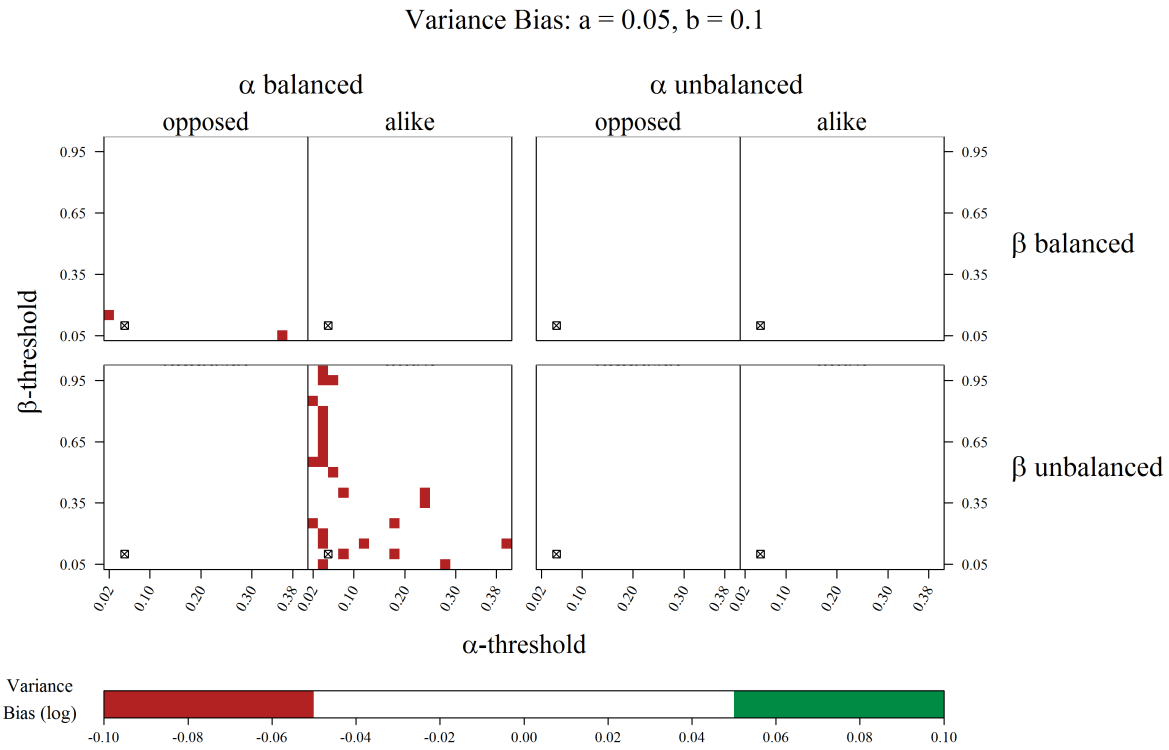

Figure S37: Bias in estimated relation of variances for the best cluster for $a = 0.05, b = 0.10$.

a=0.05, b=0.50

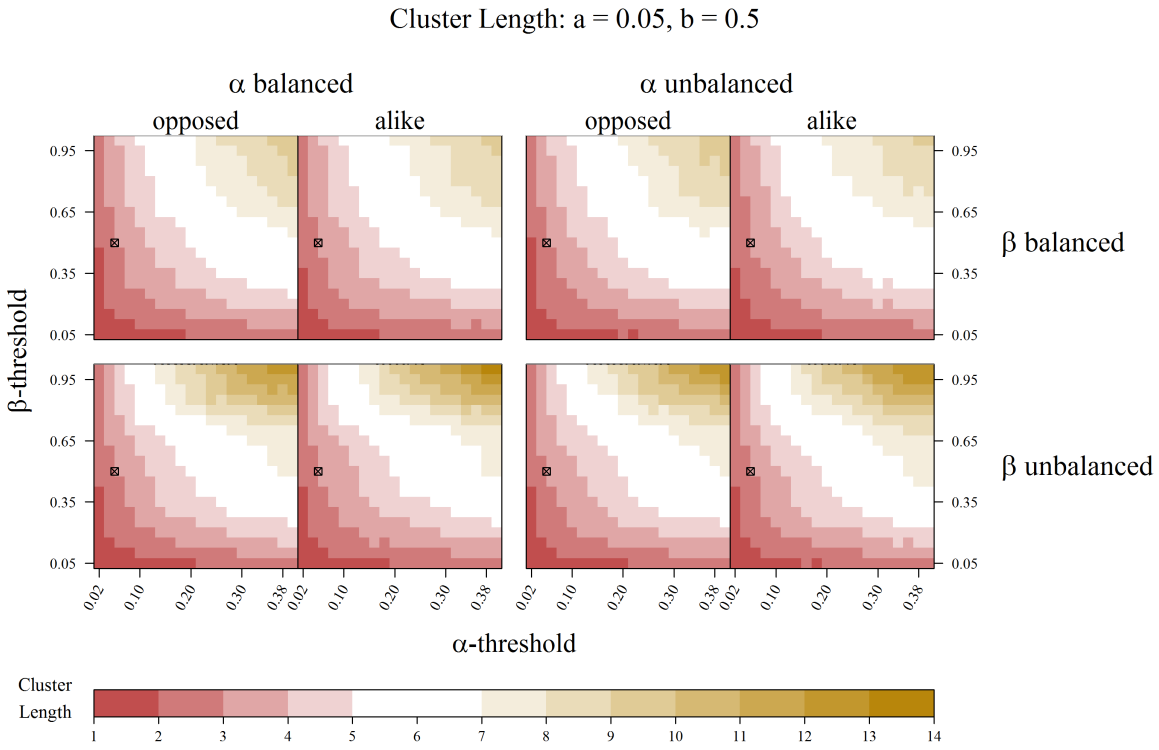

Figure S38: Cluster length for the best cluster for $a = 0.05, b = 0.50$.

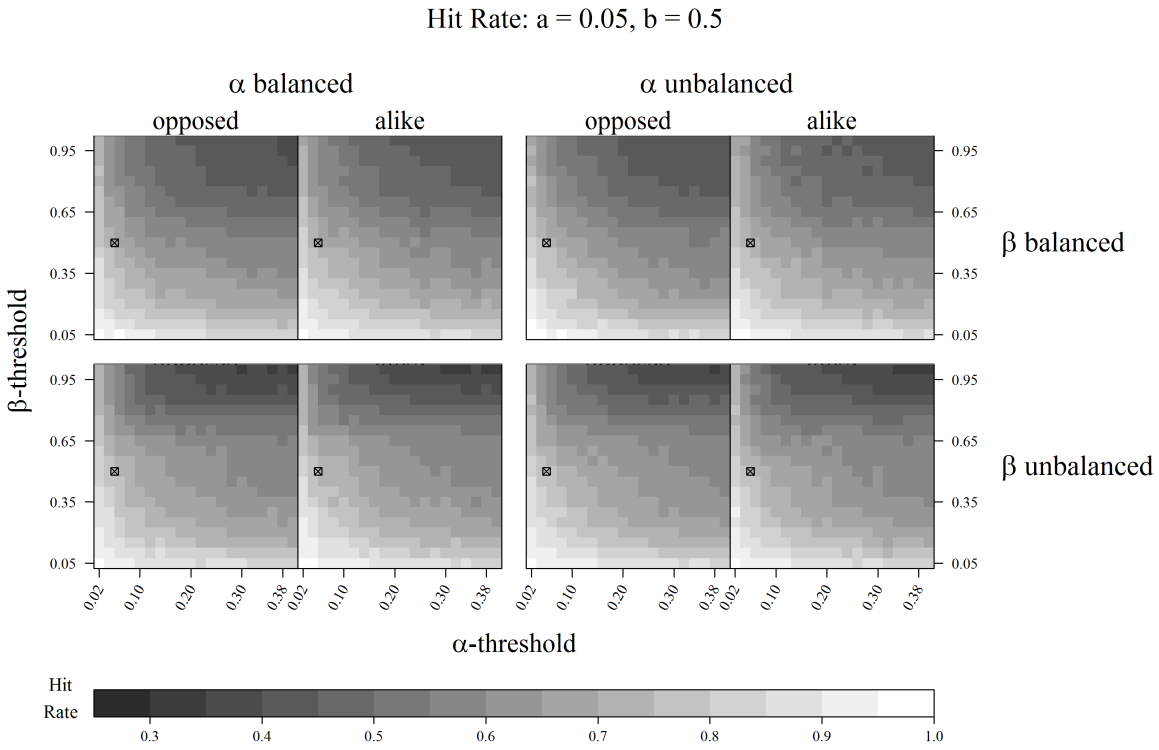

Figure S39: Hit rate for the best cluster for $a = 0.05, b = 0.50$.

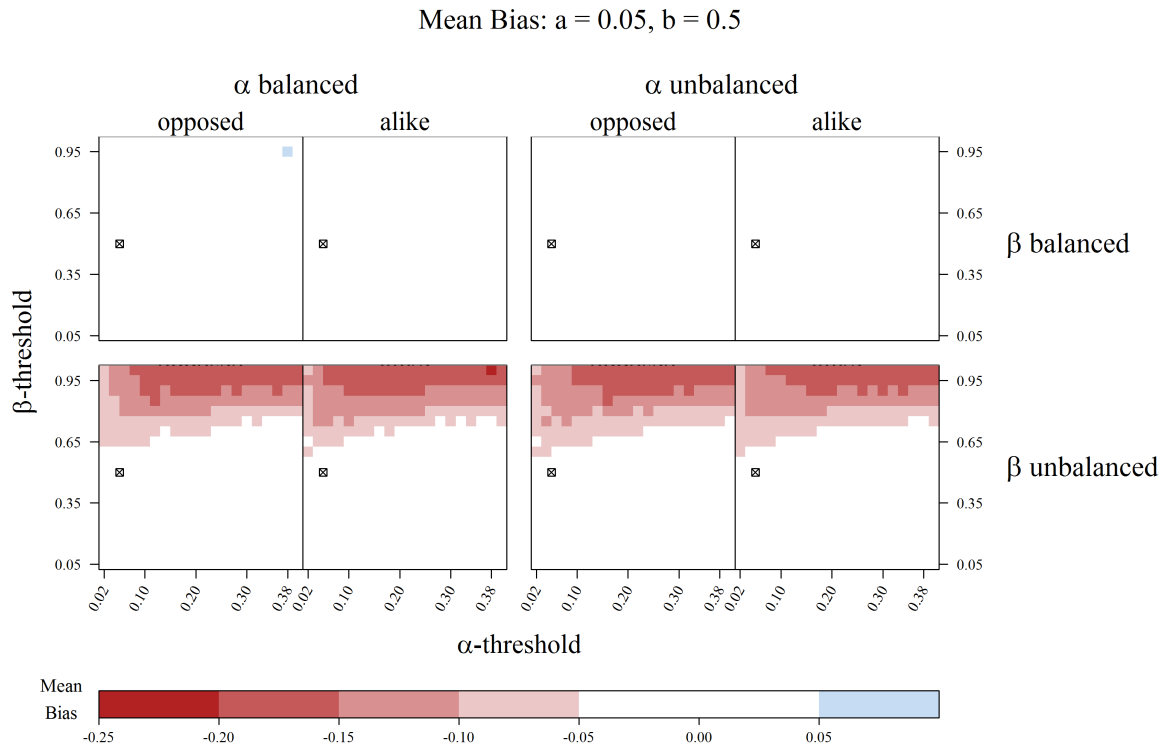

Figure S40: Bias in estimated mean difference for the best cluster for $a = 0.05, b = 0.50$.

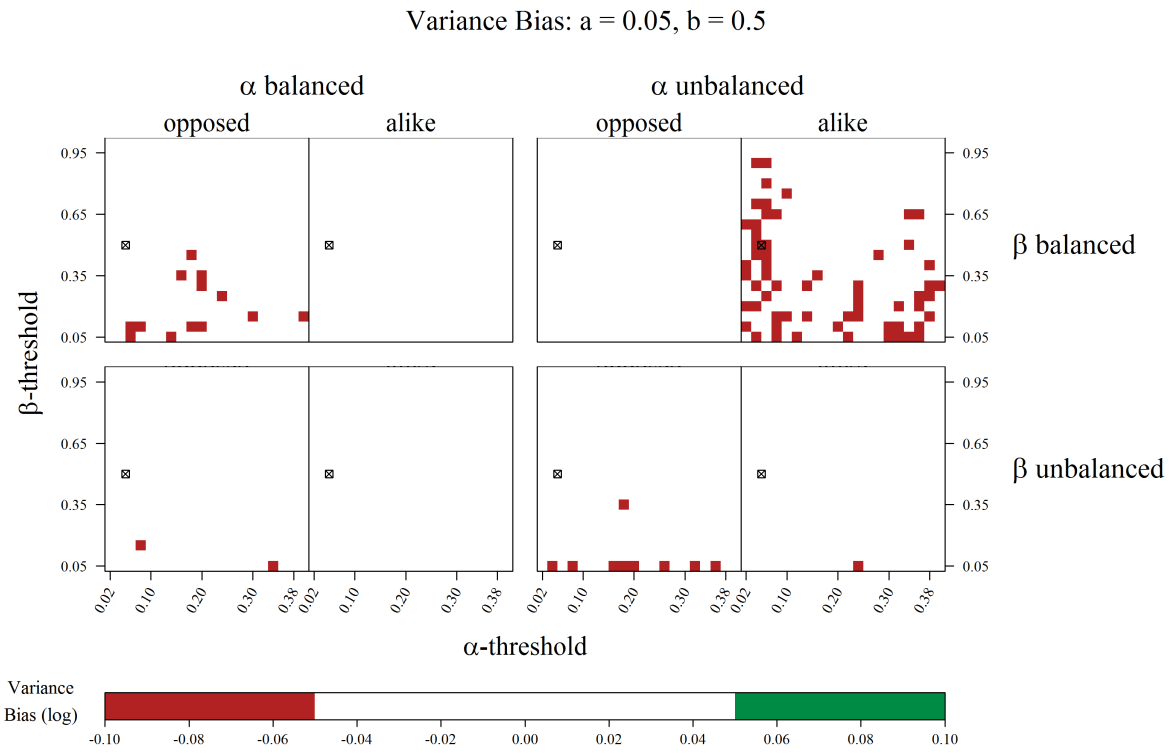

Figure S41: Bias in estimated relation of variances for the best cluster for $a = 0.05, b = 0.50$.

a=0.05, b=0.80

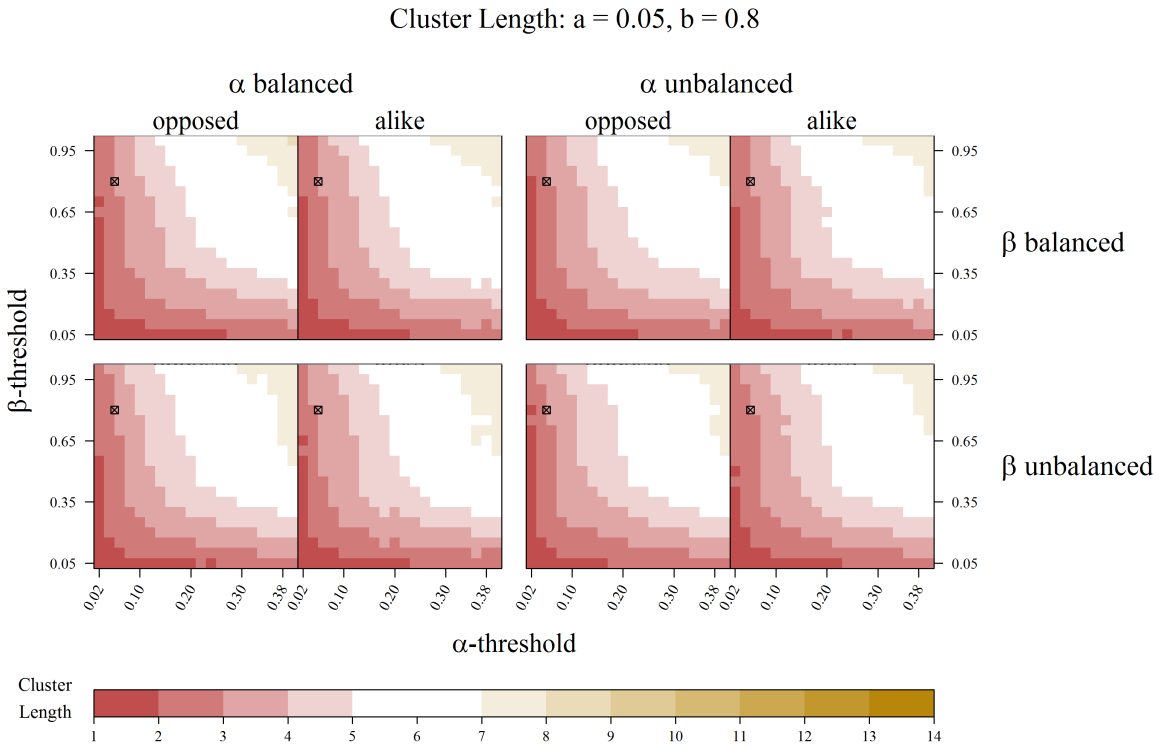

Figure S42: Cluster length for the best cluster for $a = 0.05, b = 0.80$.

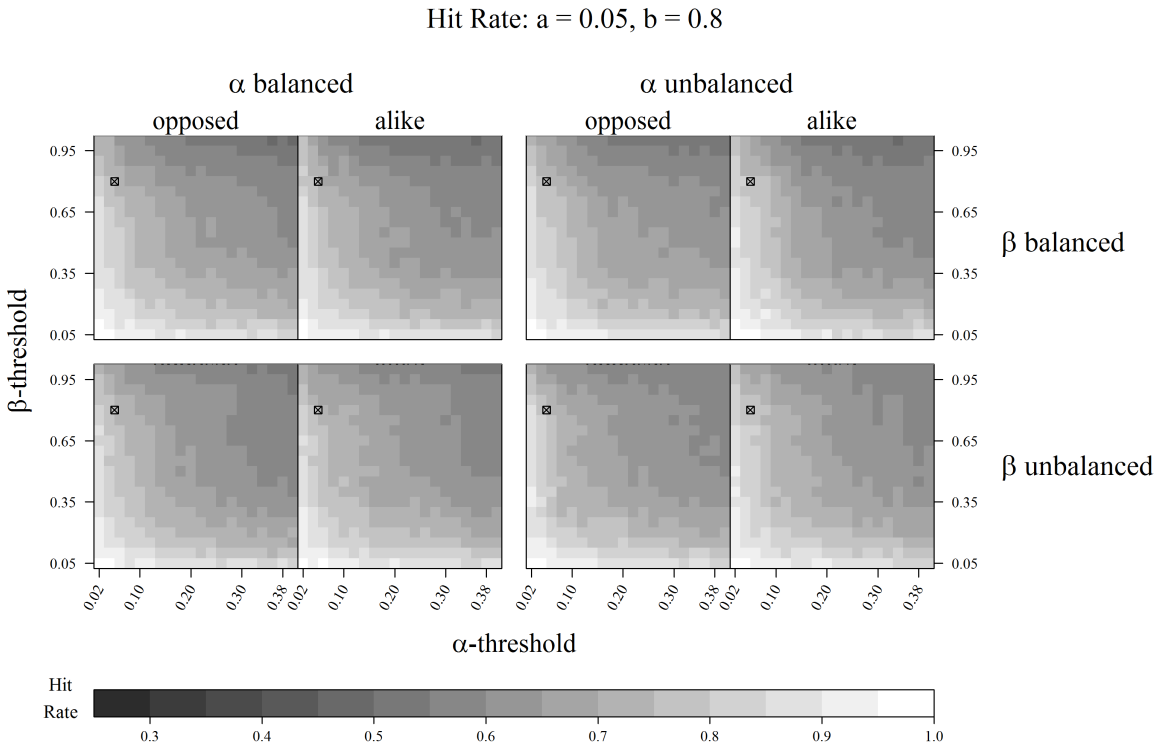

Figure S43: Hit rate for the best cluster for $a = 0.05, b = 0.80$.

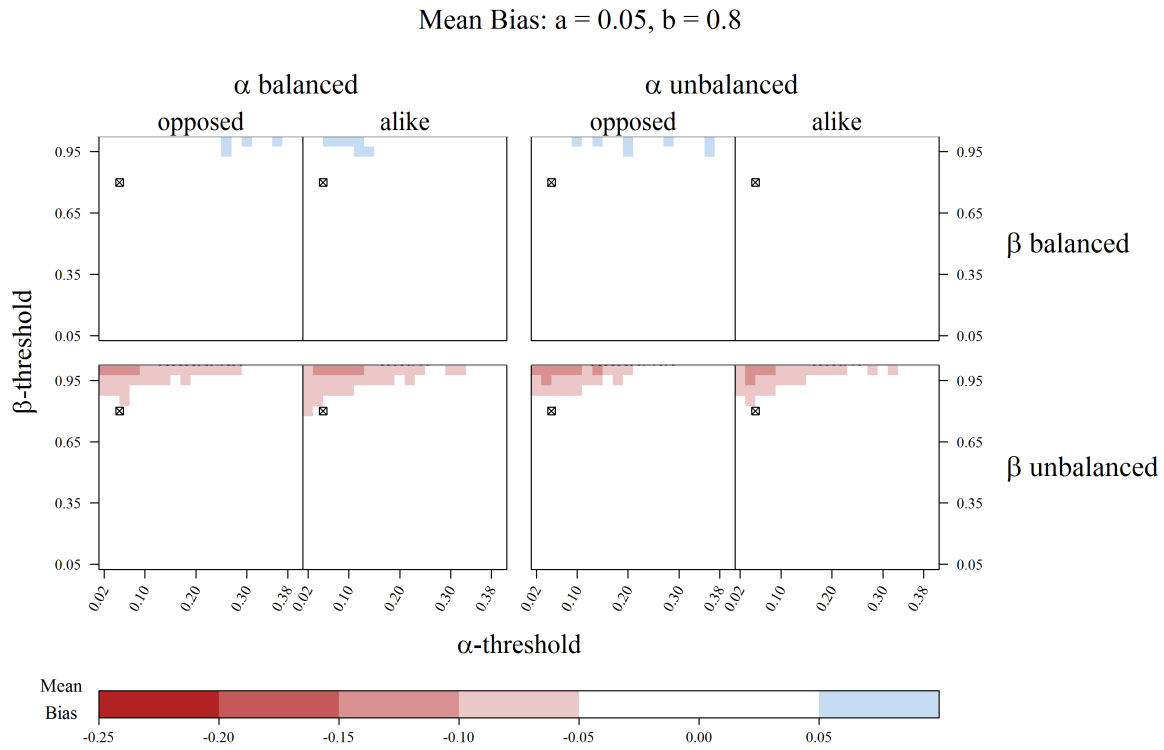

Figure S44: Bias in estimated mean difference for the best cluster for $a = 0.05, b = 0.80$.

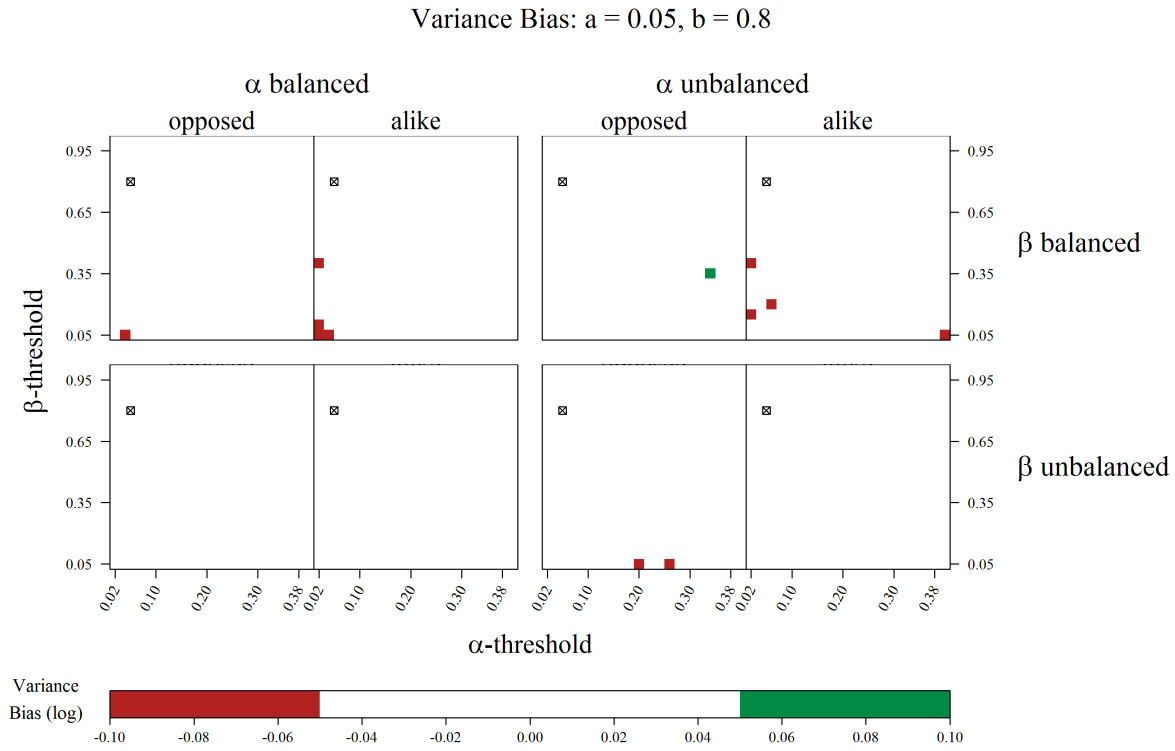

Figure S45: Bias in estimated relation of variances for the best cluster for $a = 0.05, b = 0.80$.

a=0.20, b=0.10

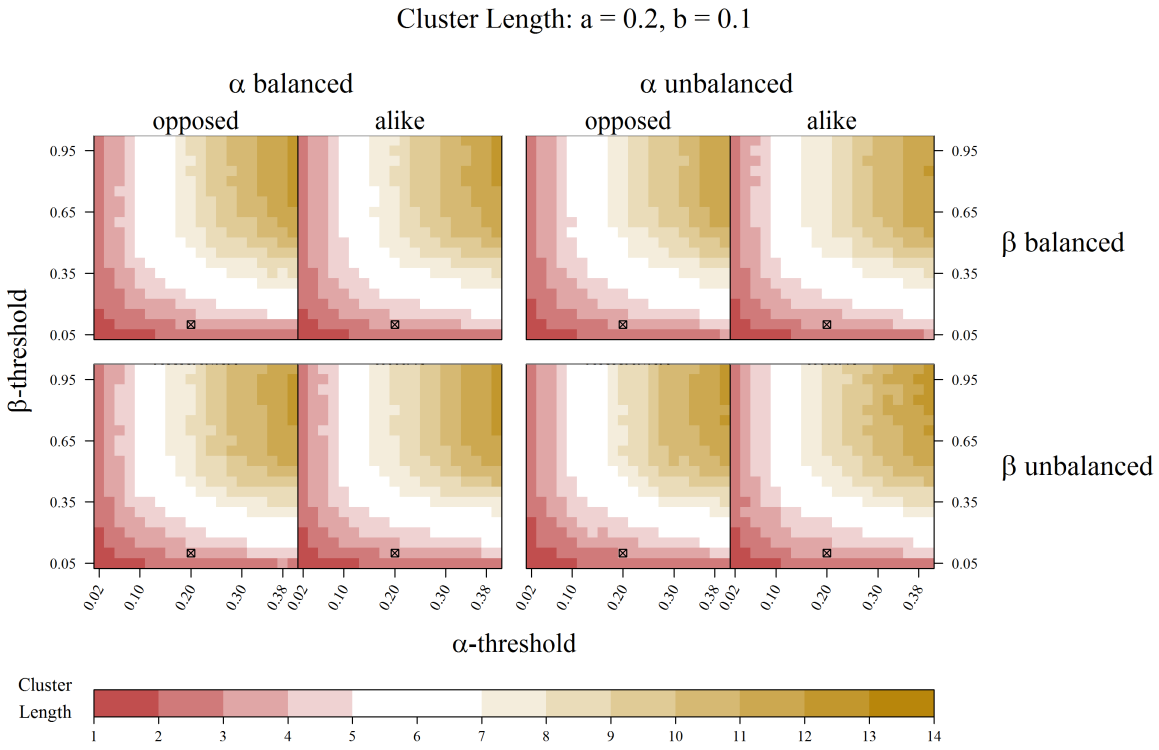

Figure S46: Cluster length for the best cluster for $a = 0.20, b = 0.10$.

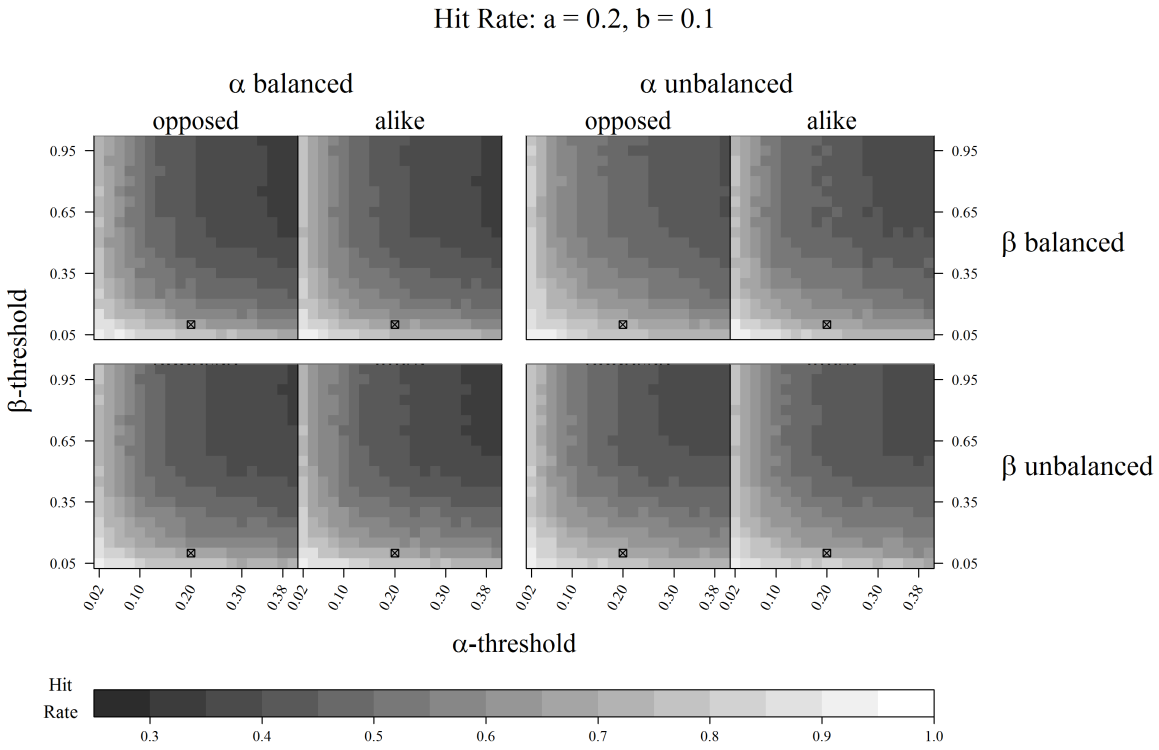

Figure S47: Hit rate for the best cluster for $a = 0.20, b = 0.10$.

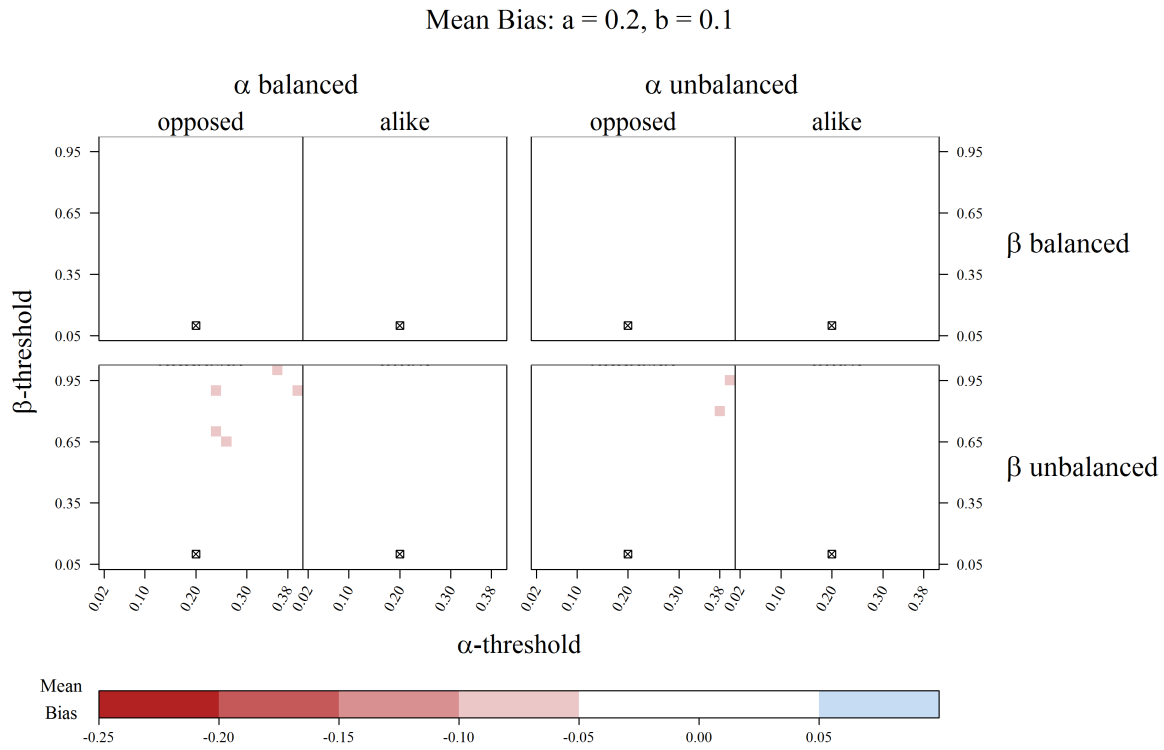

Figure S48: Bias in estimated mean difference for the best cluster for $a = 0.20, b = 0.10$.

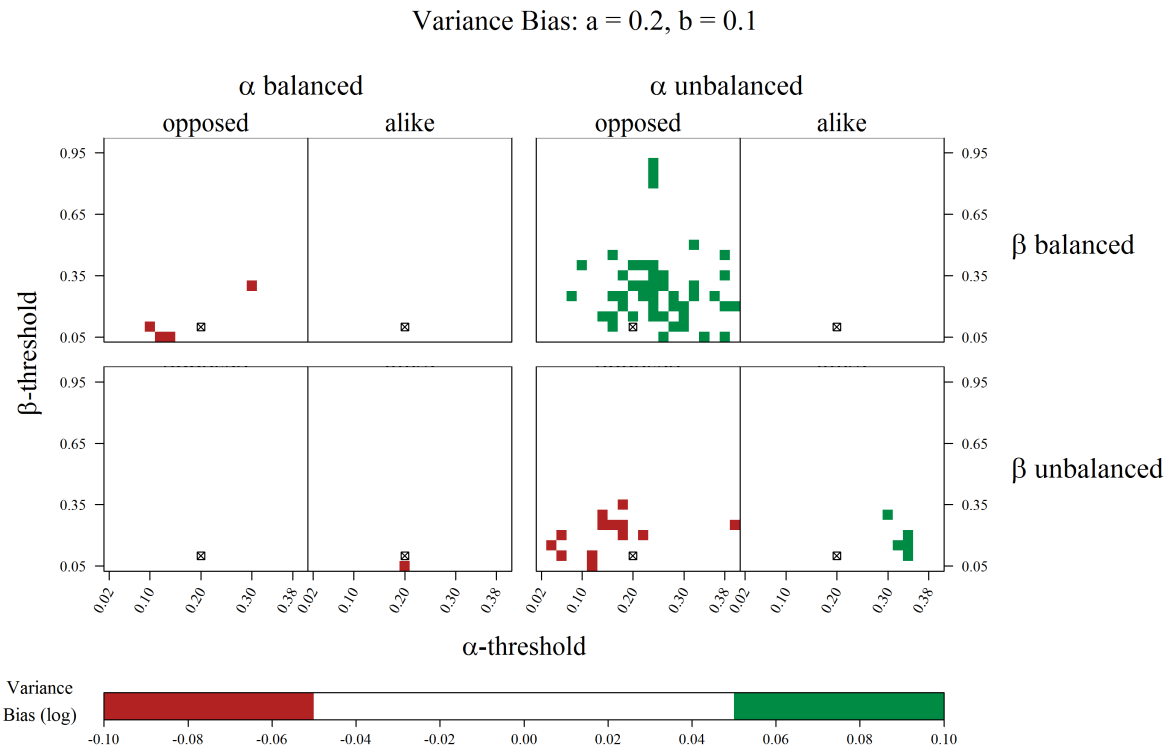

Figure S49: Bias in estimated relation of variances for the best cluster for $a = 0.20, b = 0.10$.

a=0.20, b=0.50

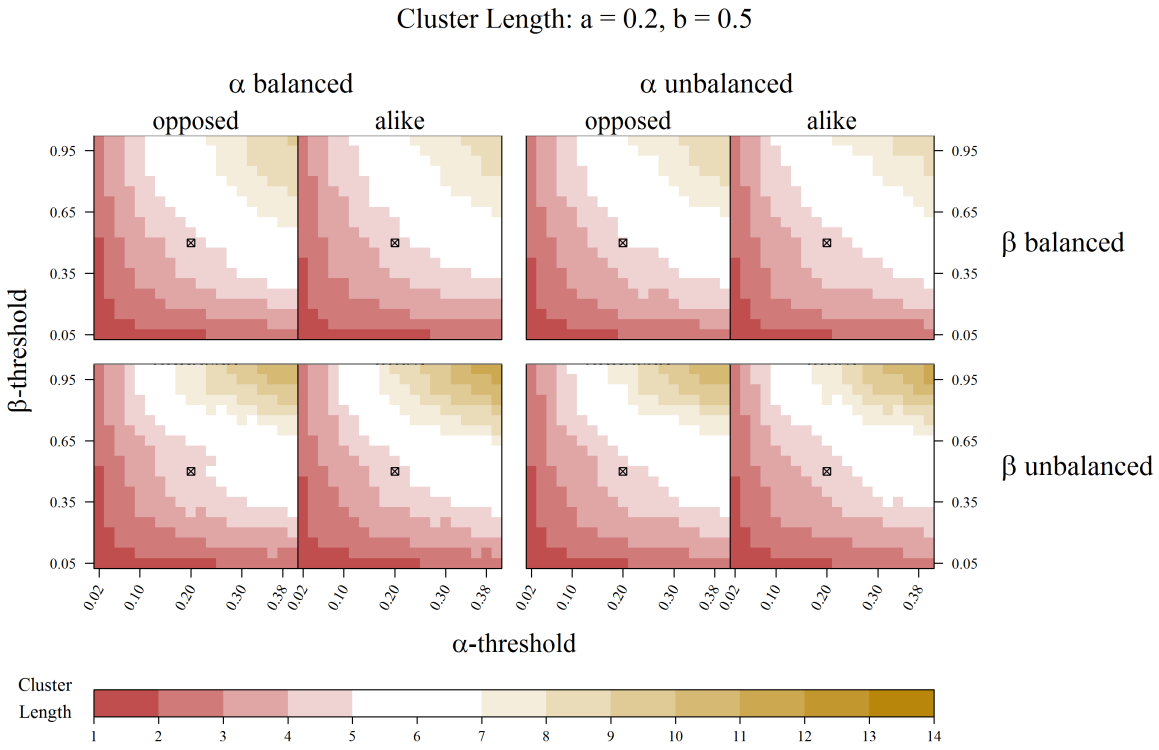

Figure S50: Cluster length for the best cluster for $a = 0.20, b = 0.50$.

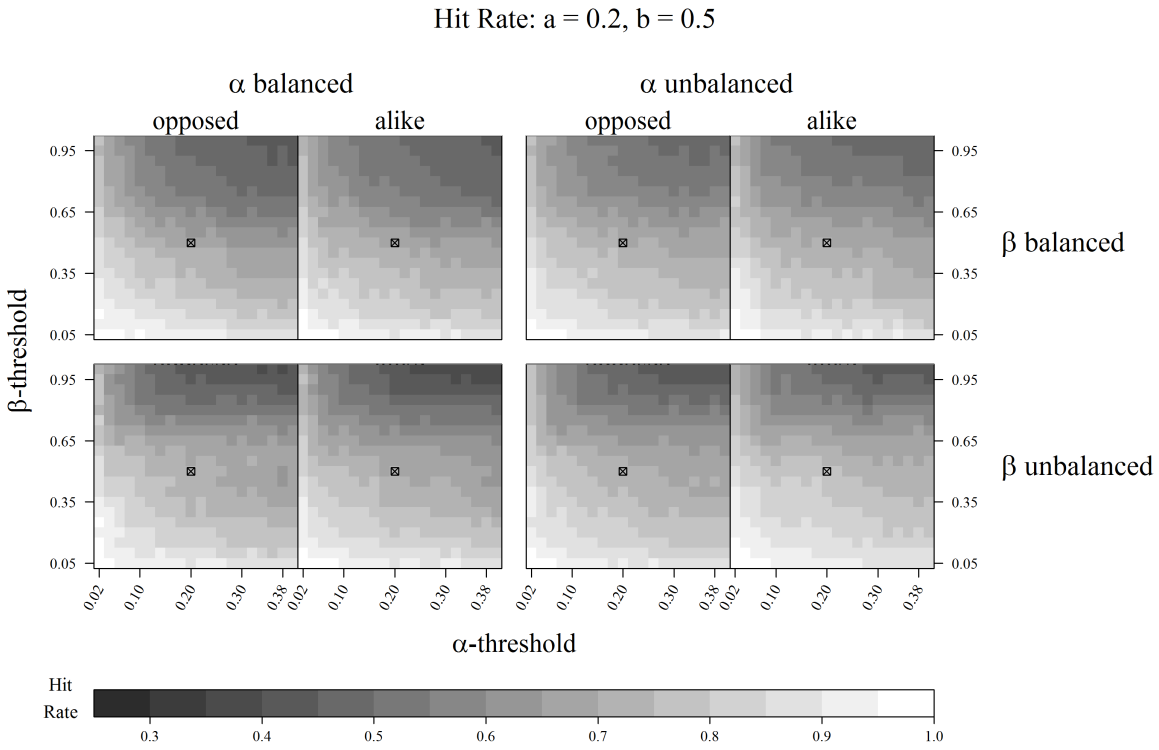

Figure S51: Hit rate for the best cluster for $a = 0.20, b = 0.50$.

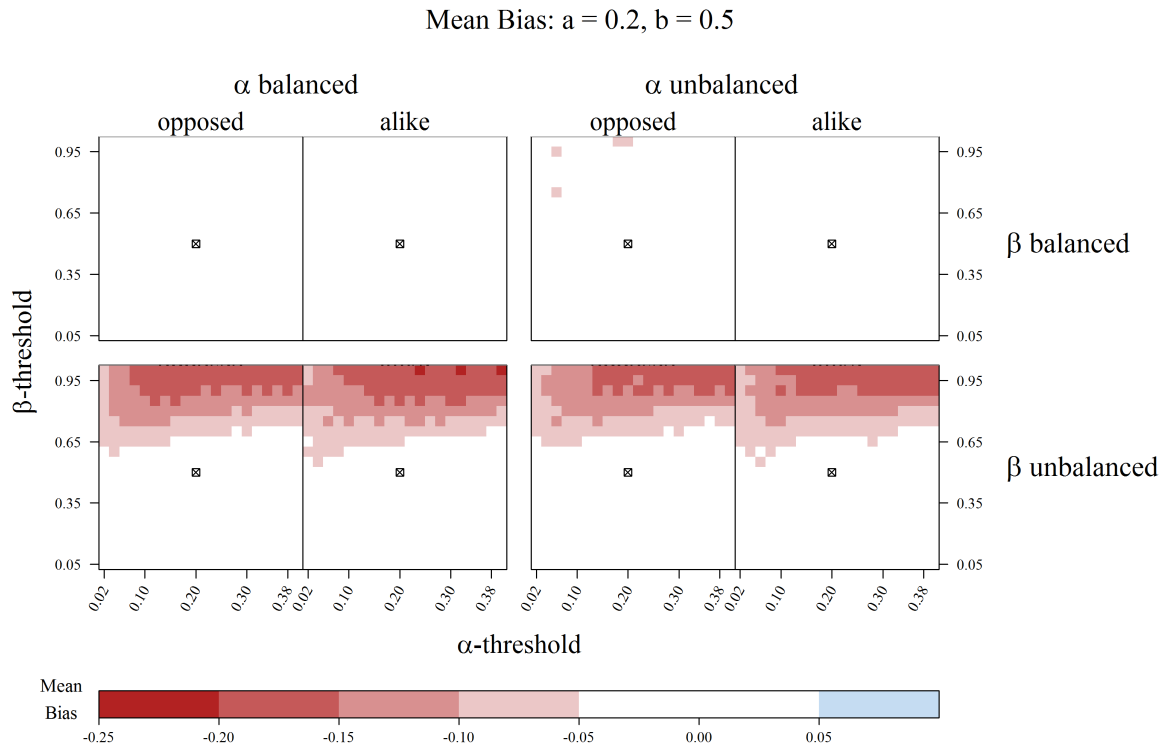

Figure S52: Bias in estimated mean difference for the best cluster for $a = 0.20, b = 0.50$.

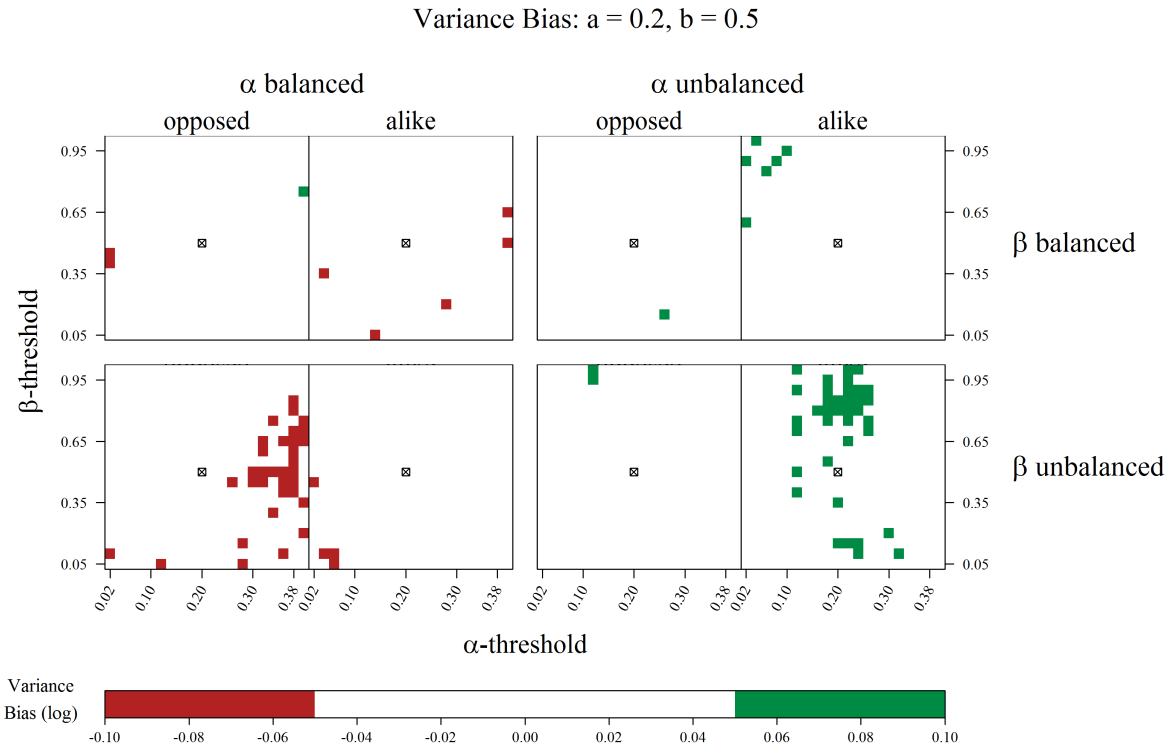

Figure S53: Bias in estimated relation of variances for the best cluster for $a = 0.20, b = 0.50$.

a=0.20, b=0.80

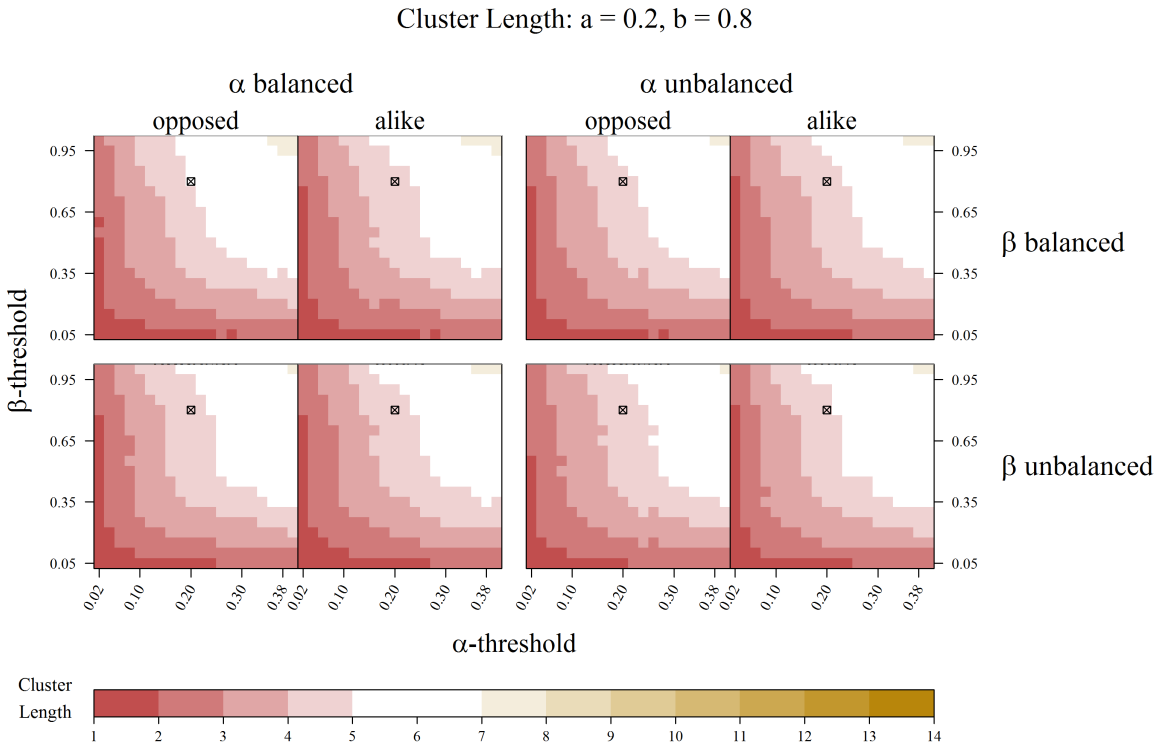

Figure S54: Cluster length for the best cluster for $a = 0.20, b = 0.80$.

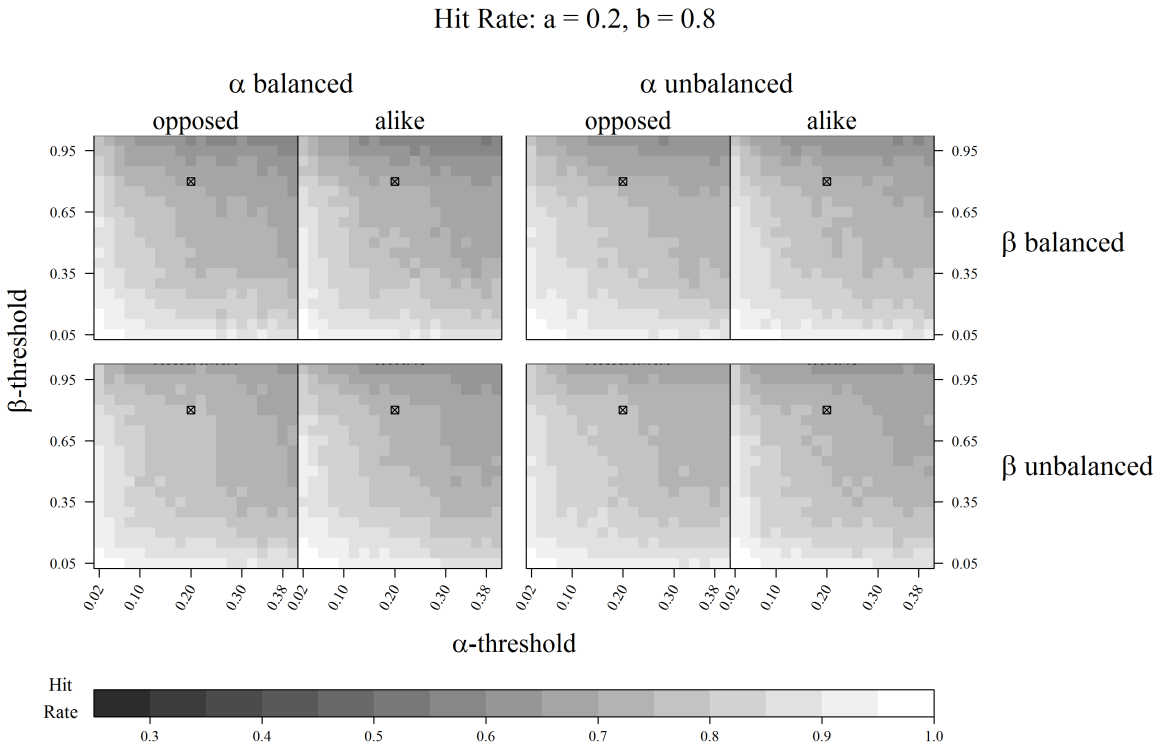

Figure S55: Hit rate for the best cluster for $a = 0.20, b = 0.80$.

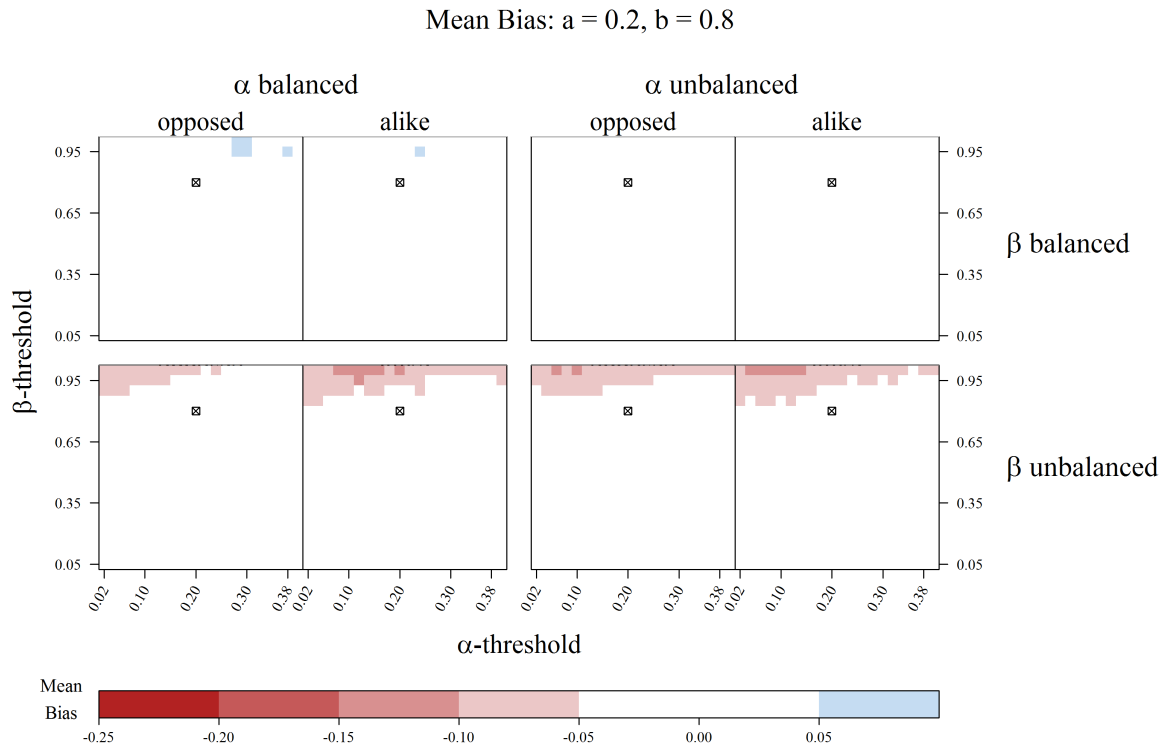

Figure S56: Bias in estimated mean difference for the best cluster for $a = 0.20, b = 0.80$.

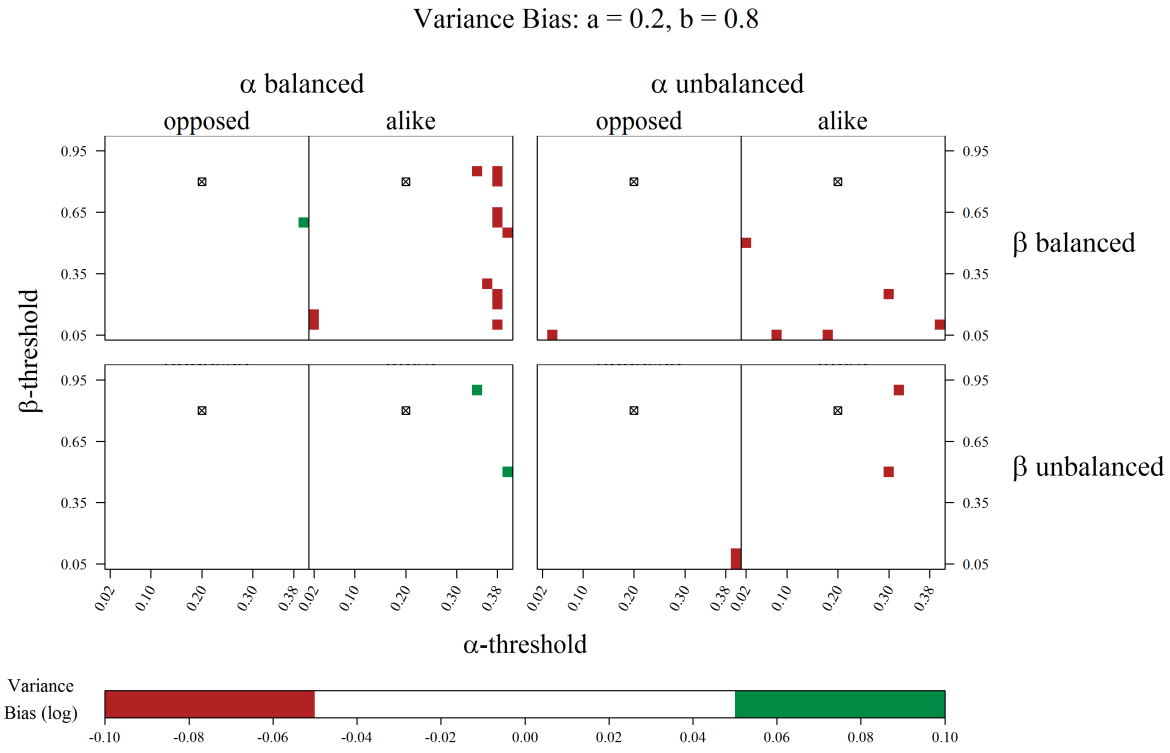

Figure S57: Bias in estimated relation of variances for the best cluster for $a = 0.20, b = 0.80$.

a=0.30, b=0.10

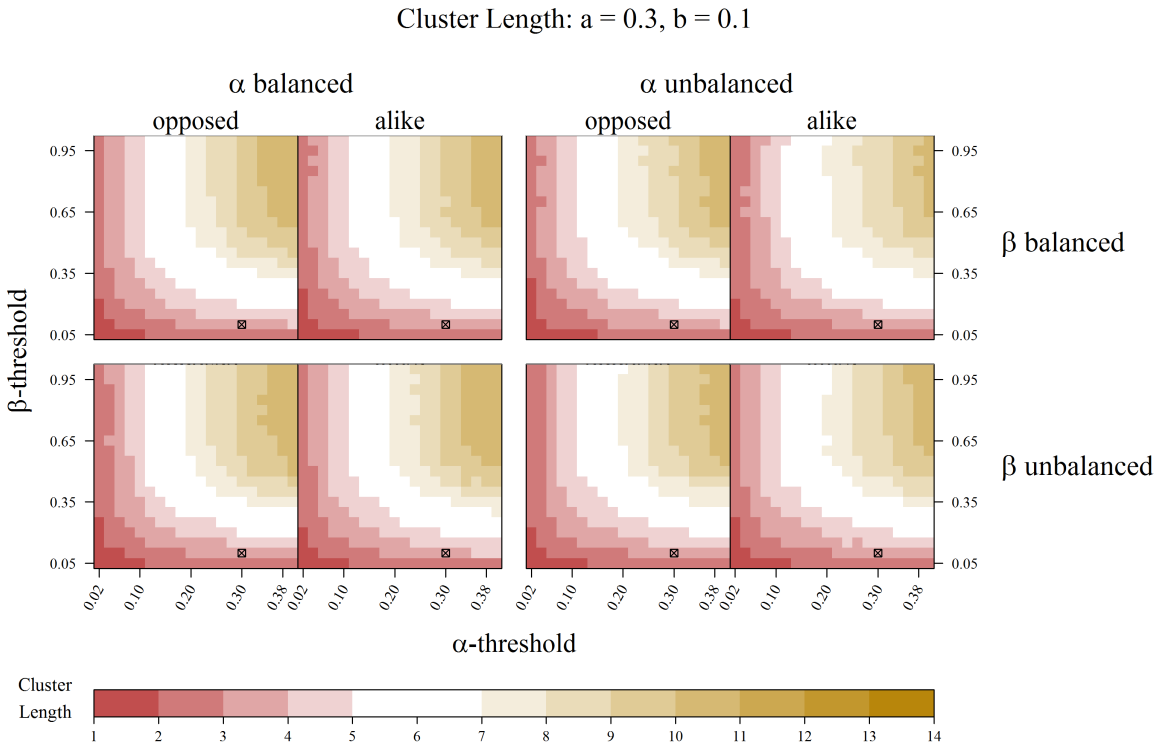

Figure S58: Cluster length for the best cluster for $a = 0.30, b = 0.10$.

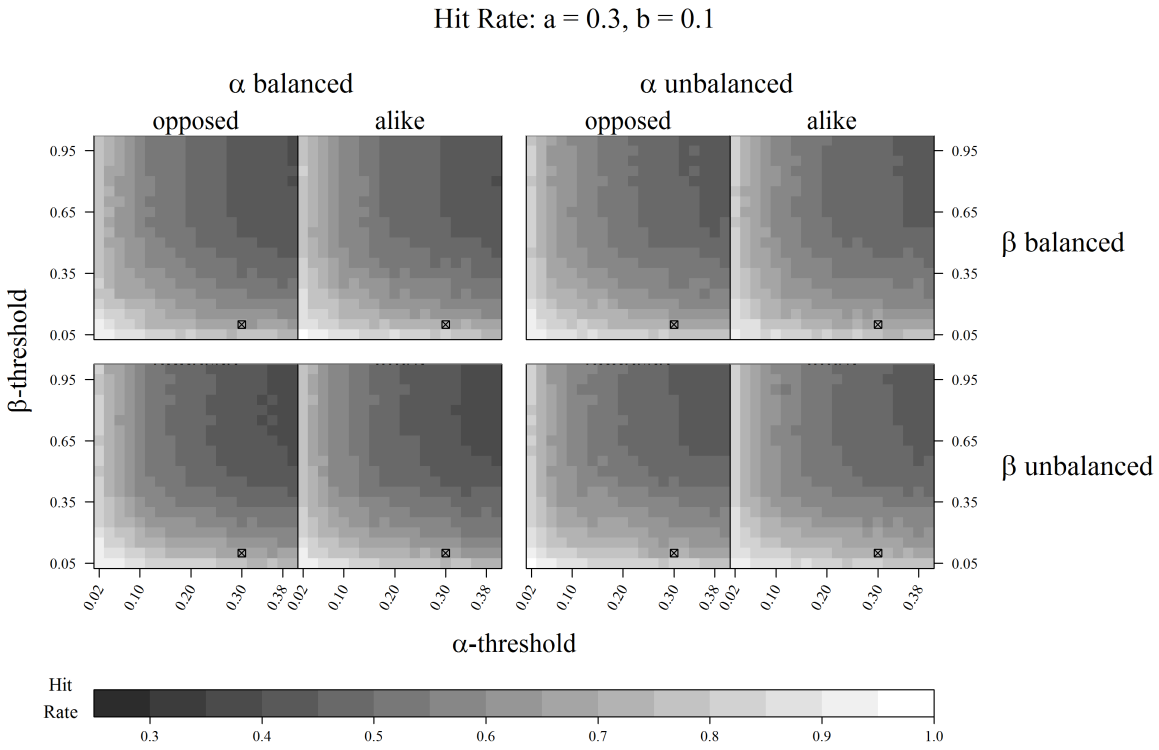

Figure S59: Hit rate for the best cluster for $a = 0.30, b = 0.10$.

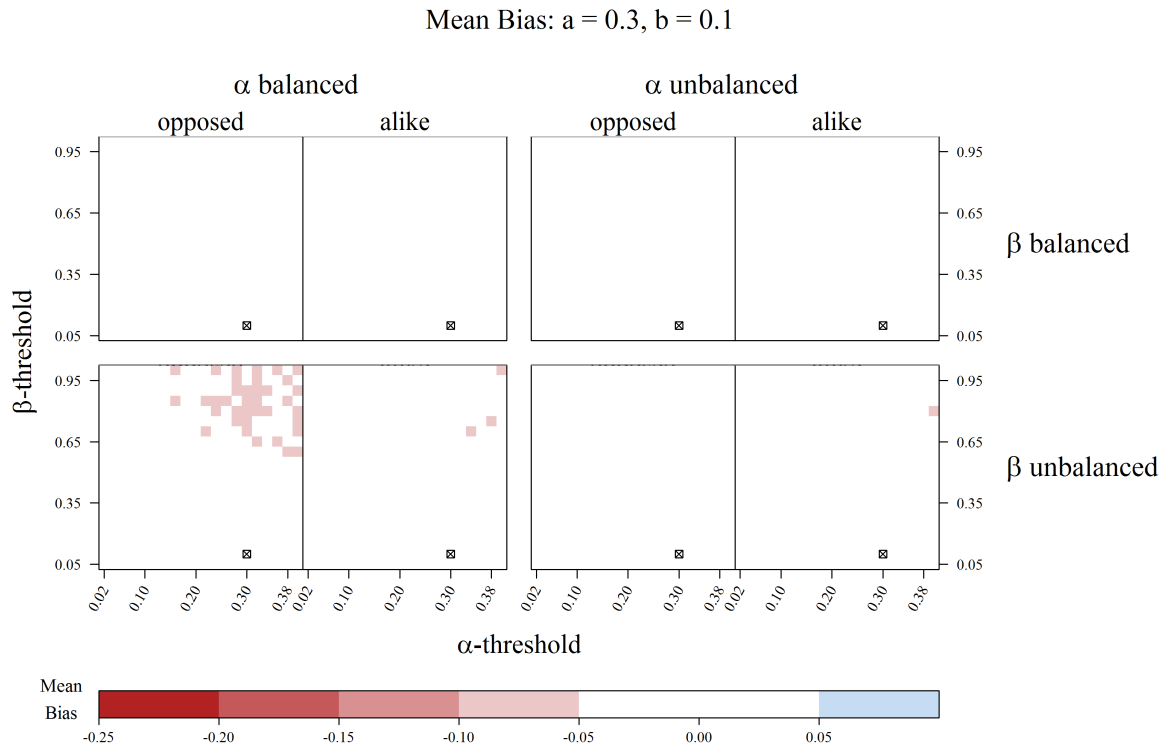

Figure S60: Bias in estimated mean difference for the best cluster for $a = 0.30, b = 0.10$.

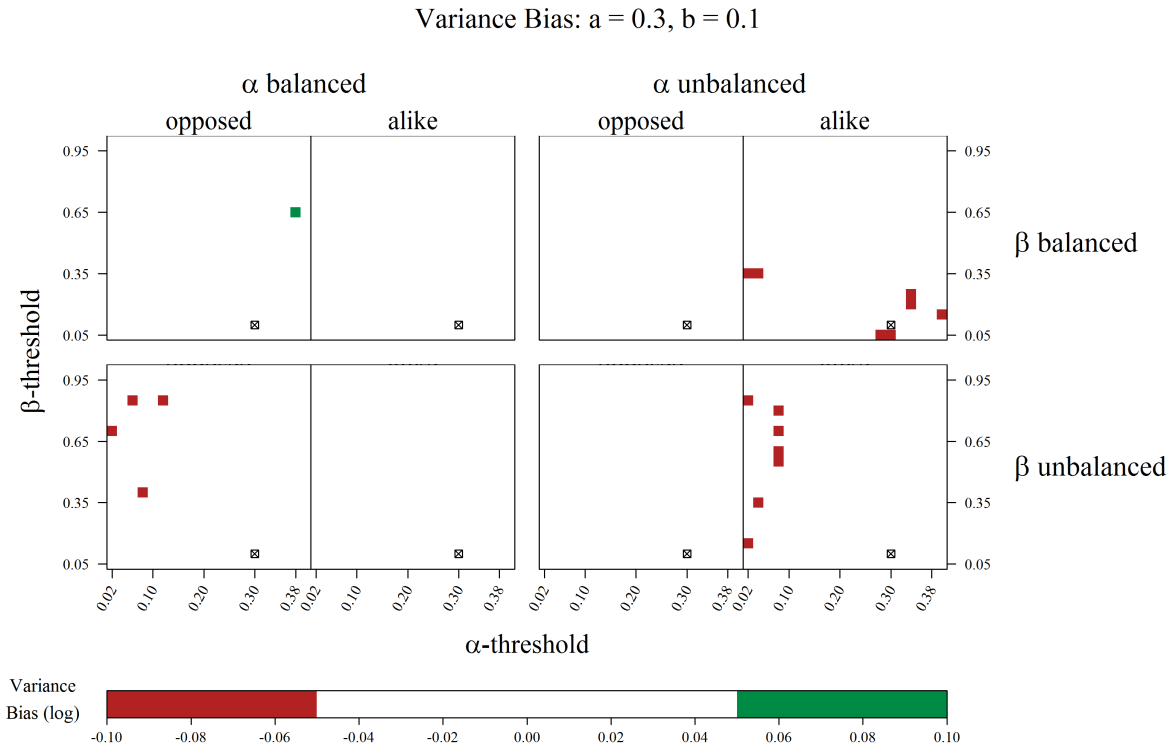

Figure S61: Bias in estimated relation of variances for the best cluster for $a = 0.30, b = 0.10$.

a=0.30, b=0.50

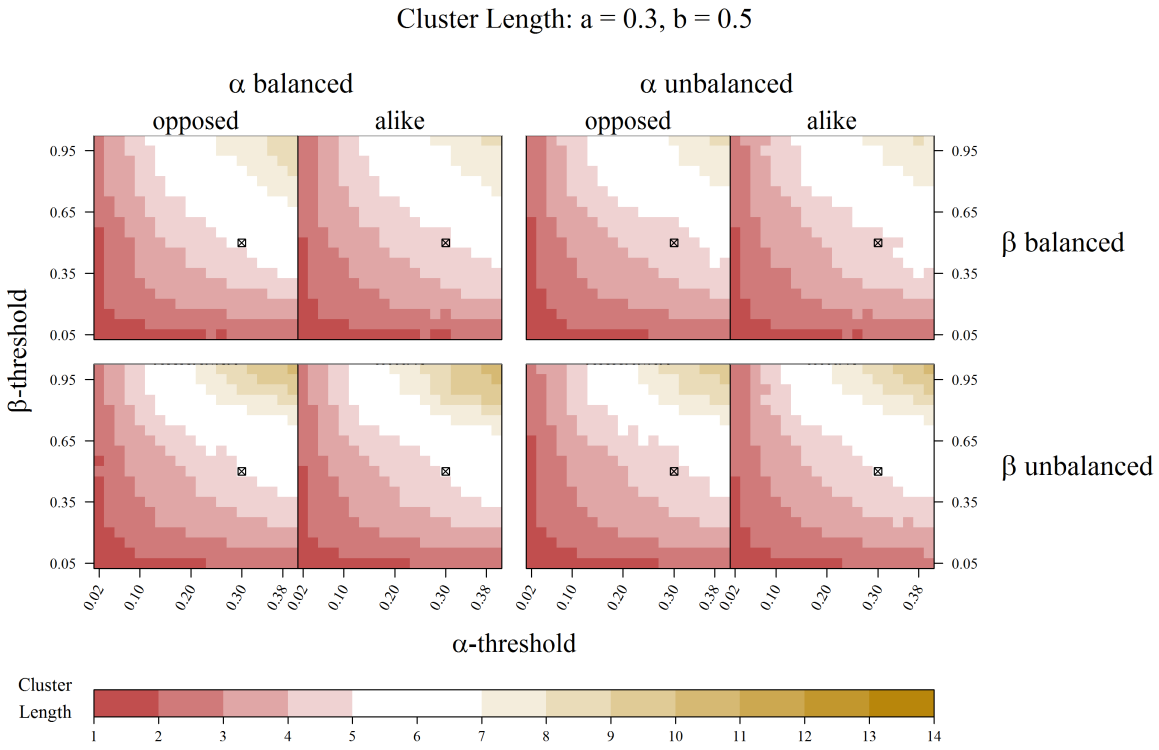

Figure S62: Cluster length for the best cluster for $a = 0.30, b = 0.50$.

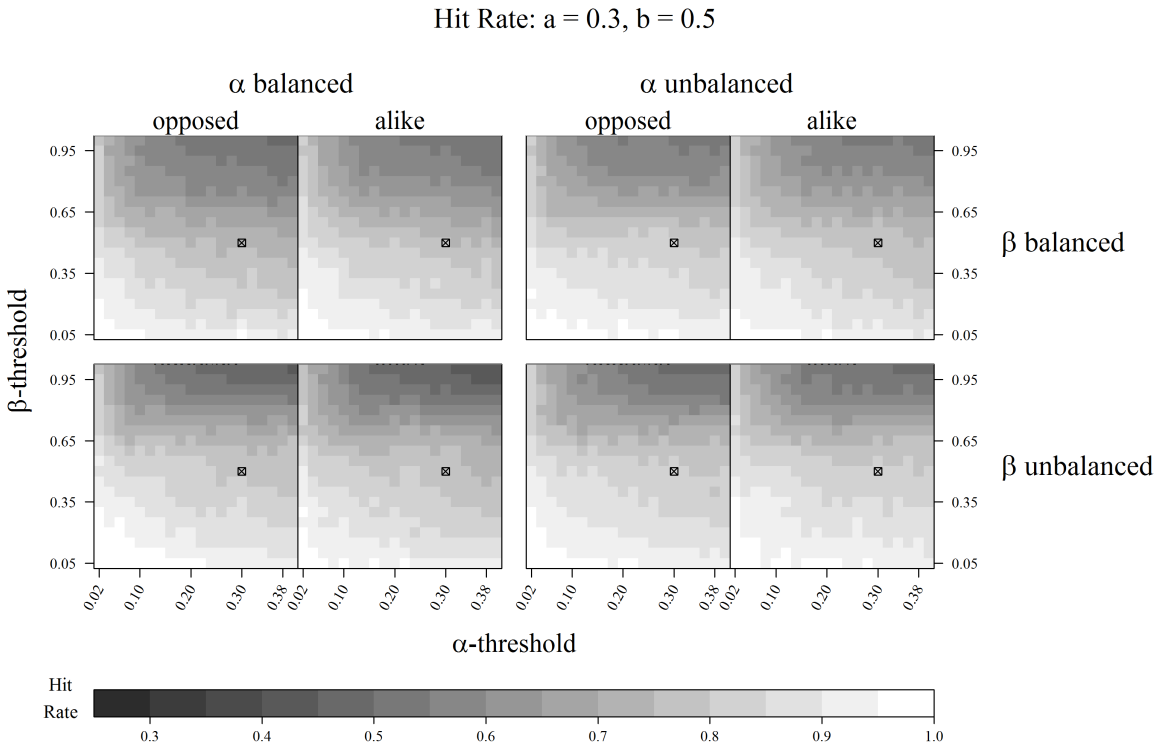

Figure S63: Hit rate for the best cluster for $a = 0.30, b = 0.50$.

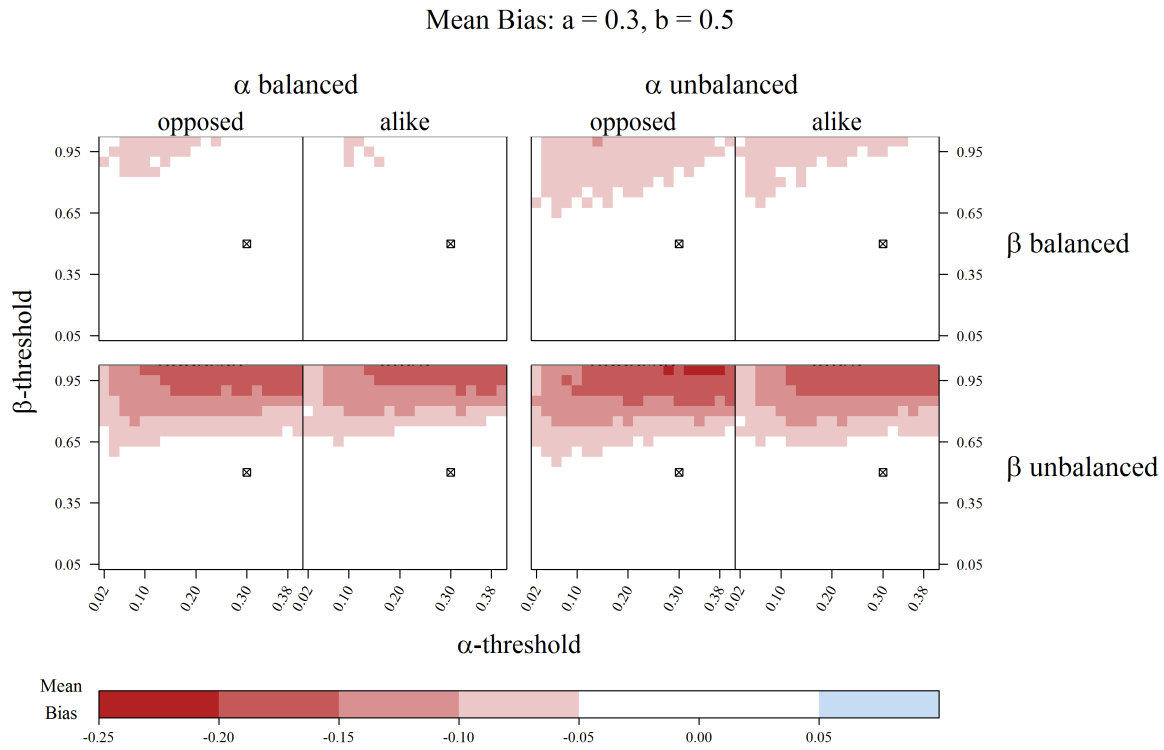

Figure S64: Bias in estimated mean difference for the best cluster for $a = 0.30, b = 0.50$.

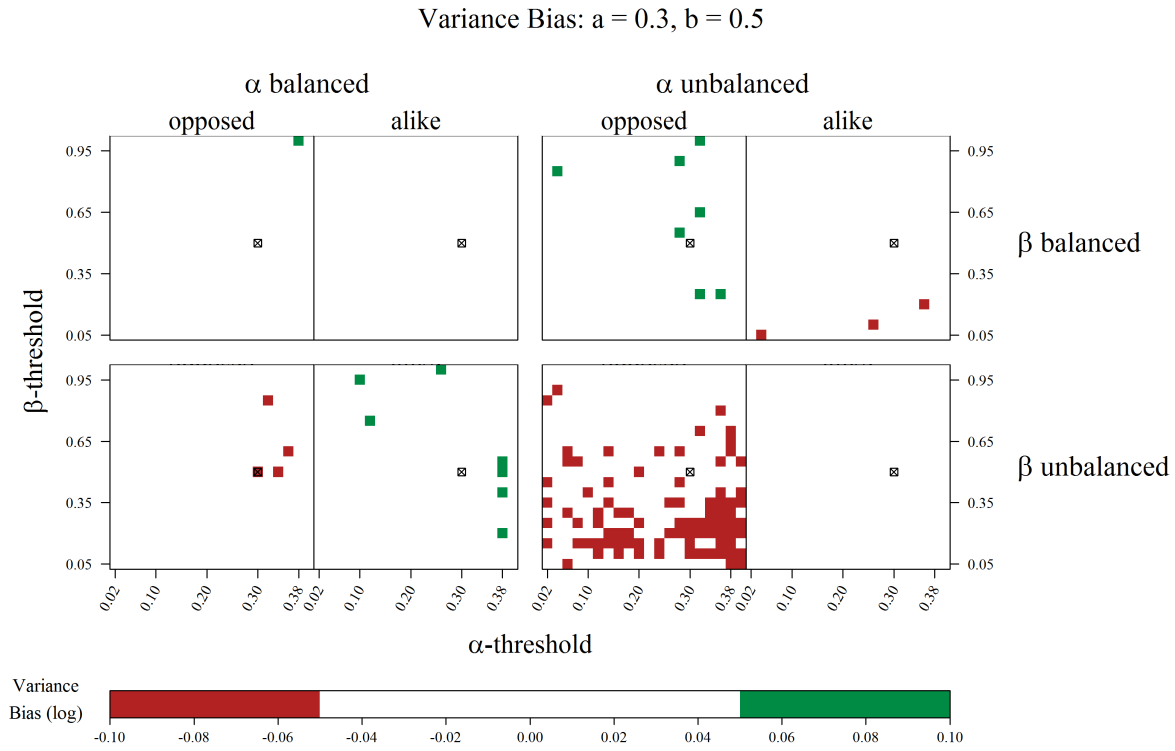

Figure S65: Bias in estimated relation of variances for the best cluster for $a = 0.30, b = 0.50$.

a=0.30, b=0.80

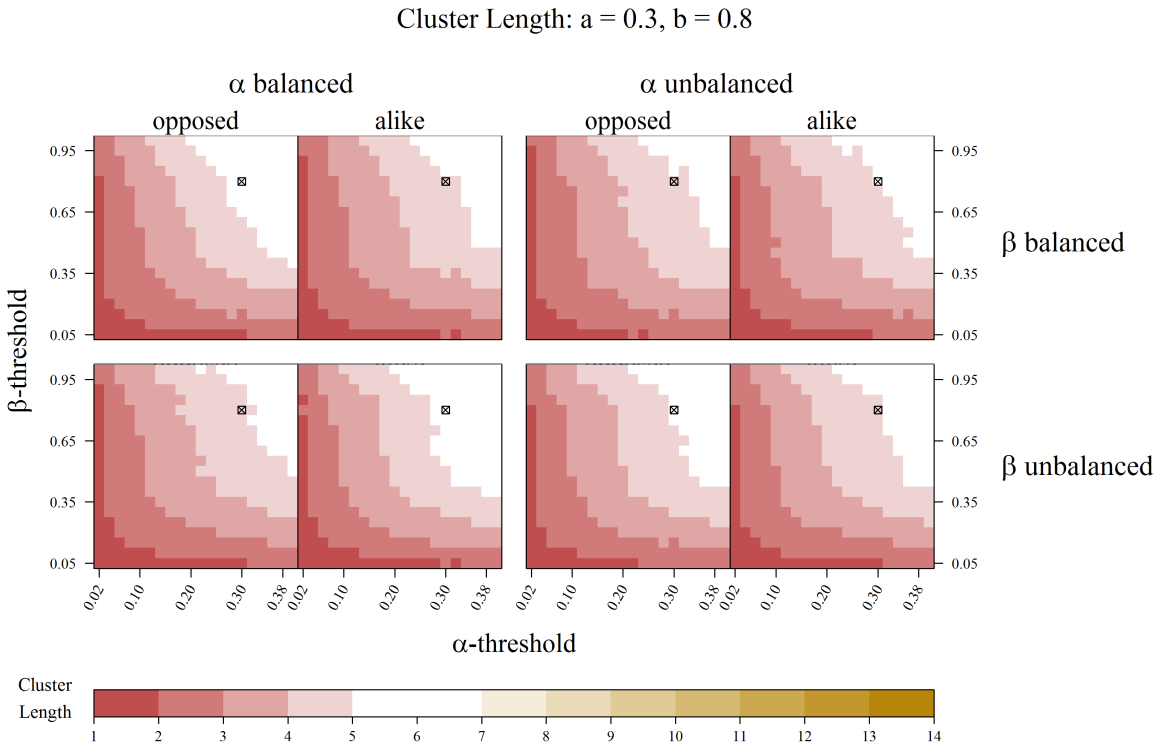

Figure S66: Cluster length for the best cluster for $a = 0.30, b = 0.80$.

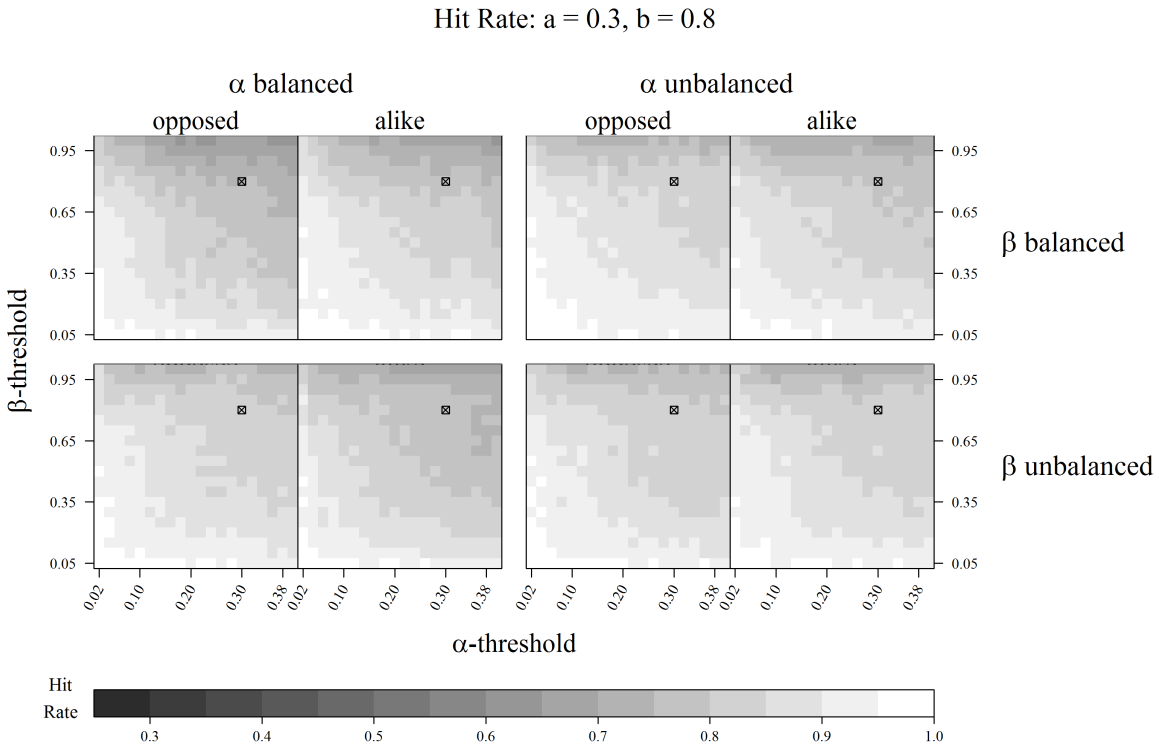

Figure S67: Hit rate for the best cluster for $a = 0.30, b = 0.80$.

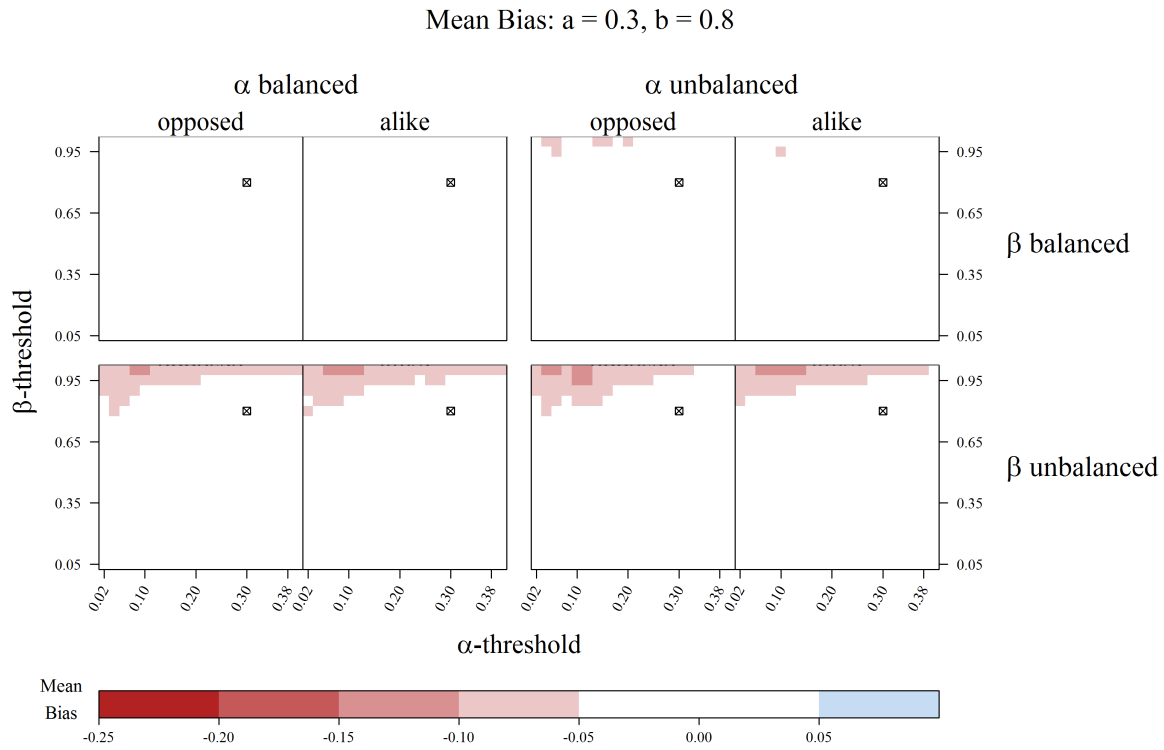

Figure S68: Bias in estimated mean difference for the best cluster for $a = 0.30, b = 0.80$.

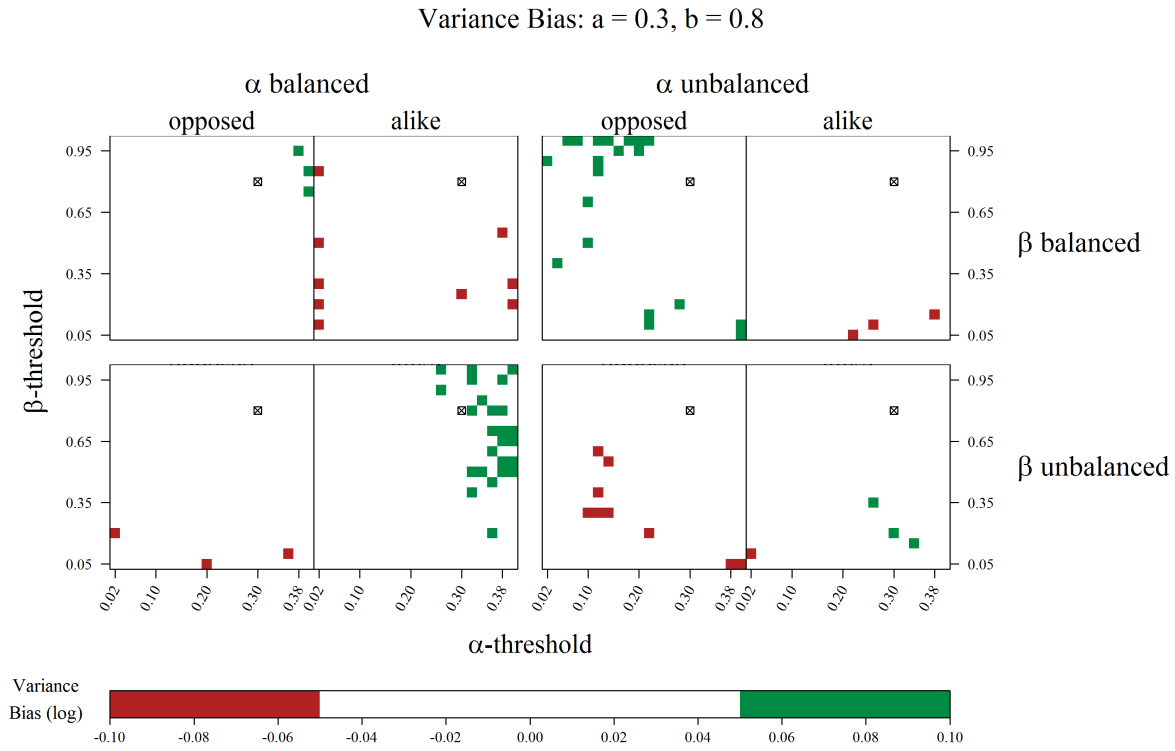

Figure S69: Bias in estimated relation of variances for the best cluster for $a = 0.30, b = 0.80$.