Supplemental material on "Improving the computational performance of TCLUST through ensemble initialization"

Pedro C. Álvarez-Esteban, Luis A. García-Escudero,
Agustín Mayo-Iscar, Javier Crespo-Guerrero

1 R Code for simula_tclust

```
simula_tclust <- function(n, p = 4, G = 3, scenario = 2, balanced = 1, symmetry = 1)
{
  if (G != 3 & G != 6)
    stop("Invalid G")
  if (p < 2)
    stop("Invalid p")
  if (symmetry != 1 & symmetry != 2)
    stop("Invalid symmetry")
  if (G == 6) {
    nn <- n/2
  }
  else {
    nn <- n
  }
  if (balanced == 1) {
    nn1 = nn2 = nn3 = ceiling(nn * 0.3)
  }
  else {
    nn1 = ceiling(nn * 0.25)
    nn2 = ceiling(nn * 0.3)
    nn3 = ceiling(nn * 0.35)
  }
 if (symmetry == 1 ){
   cc1 <- -20
   cc2 <- -50
   } else {
   cc1 <- 0
   cc2 <- 0
 }

  nh <- nn1 + nn2 + nn3
  Y <- matrix(NA, ncol = p, nrow = n)
  if (scenario == 1) {
    if (G == 3) {
      Y[1:nn1, ] <- MASS::mvrnorm(nn1, c(4, 4), matrix(c(1, 0, 0, 1), nrow = 2))
      Y[(nn1 + 1):(nn1 + nn2), ] <- MASS::mvrnorm(nn2, c(-4, 4), matrix(c(1, 0, 0, 1), nrow = 2))
      Y[(nn1 + nn2 + 1):(nn1 + nn2 + nn3), ] <- MASS::mvrnorm(nn3, c(0, 0), matrix(c(1, 0, 0, 1), nrow = 2))
      Y[(nh + 1):n, ] <- matrix(runif(p * (n - nh), cc1, 20), ncol = p)
      if (p > 2) {
        Y[1:n, 3:p] <- MASS::mvrnorm(n, rep(0, p - 2), diag(p - 2))
      }
    }
    else {
      Y[1:nn1, ] <- MASS::mvrnorm(nn1, c(4, 4), matrix(c(1, 0, 0, 1), nrow = 2))
      Y[(nn1 + 1):(nn1 + nn2), ] <- MASS::mvrnorm(nn2, c(-4, 4), matrix(c(1, 0, 0, 1), nrow = 2))
      Y[(nn1 + nn2 + 1):(nn1 + nn2 + nn3), ] <- MASS::mvrnorm(nn3, c(0, 0), matrix(c(1, 0, 0, 1), nrow = 2))
      Y[(nh + 1):(nh + nn1), ] <- MASS::mvrnorm(nn1, c(24, 0), matrix(c(1, 0, 0, 1), nrow = 2))
      Y[(nh + nn1 + 1):(nh + nn1 + nn2), ] <- MASS::mvrnorm(nn2,  c(16, 0), matrix(c(1, 0, 0, 1), nrow = 2))
      Y[(nh + nn1 + nn2 + 1):(nh + nn1 + nn2 + nn3), ] <- MASS::mvrnorm(nn3, c(20, 4), matrix(c(1, 0, 0, 1), nrow = 2))
      Y[(2 * nh + 1):n, ] <- matrix(runif((n - 2 * nh) * p, cc1, 40), ncol = p)
```

```
      if (p > 2) {
        Y[1:n, 3:p] <- MASS::mvrnorm(n, rep(0, p - 2), diag(p - 2))
      }
    }
  }
  if (scenario == 2) {
    alp1 <- runif(1, 0, 2 * pi)
    u1 <- matrix(c(cos(alp1), sin(alp1), -sin(alp1), cos(alp1)), ncol = 2)
    alp2 <- runif(1, 0, 2 * pi)
    u2 <- matrix(c(cos(alp2), sin(alp2), -sin(alp2), cos(alp2)), ncol = 2)
    if (G == 3) {
      Y[1:nn1, ] <- MASS::mvrnorm(nn1, c(20, 20), t(u1) %*%  diag(c(1, 9^2)) %*% u1)
      Y[(nn1 + 1):(nn1 + nn2), ] <- MASS::mvrnorm(nn2, c(-20, -20), t(u1) %*% diag(c(9^2, 1)) %*%
          u1)
      Y[(nn1 + nn2 + 1):(nn1 + nn2 + nn3), ] <- MASS::mvrnorm(nn3,  c(0, 0), t(u1) %*%
          diag(c(3^2, 3^2)) %*% u1)
      Y[(nn1 + nn2 + nn3 + 1):n, ] <- matrix(runif(p * (n - nh), cc2, 50), ncol = p)
      if (p > 2) {
        Y[1:n, 3:p] <- MASS::mvrnorm(n, rep(0, p - 2), diag(p - 2))
      }
    }
    else {
      Y[1:nn1, ] <- MASS::mvrnorm(nn1, c(20, 20), t(u1) %*%  diag(c(1, 9^2)) %*% u1)
      Y[(nn1 + 1):(nn1 + nn2), ] <- MASS::mvrnorm(nn2, c(-20, -20), t(u1) %*% diag(c(9^2, 1)) %*%
          u1)
      Y[(nn1 + nn2 + 1):(nn1 + nn2 + nn3), ] <- MASS::mvrnorm(nn3, c(0, 0), t(u1) %*% diag(c(3^2,
          3^2)) %*% u1)
      Y[(nh + 1):(nh + nn1), ] <- MASS::mvrnorm(nn1, c(40, 20), t(u2) %*% diag(c(1, 9^2)) %*% u2)
      Y[(nh + nn1 + 1):(nh + nn1 + nn2), ] <- MASS::mvrnorm(nn2, c(0, -20), t(u2) %*% diag(c(9^2,
          1)) %*% u2)
      Y[(nh + nn1 + nn2 + 1):(nh + nn1 + nn2 + nn3), ] <- MASS::mvrnorm(nn3,  c(20, 0), t(u2) %*%
          diag(c(3^2, 3^2)) %*% u2)
      Y[(2 * nh + 1):n, ] <- matrix(runif((n - 2 * nh) * p, cc2-10, 80), ncol = p)
      if (p > 2) {
        Y[1:n, 3:p] <- MASS::mvrnorm(n, rep(0, p - 2), diag(p - 2))
      }
    }
  }
  if (scenario == 3) {
    if (G == 3) {
      Y[1:nn1, ] <- MASS::mvrnorm(nn1, c(5, 5), matrix(c(1, 0, 0, 1), nrow = 2))
      Y[(nn1 + 1):(nn1 + nn2), ] <- MASS::mvrnorm(nn2, c(-15, -15), matrix(c(1, 0, 0, 1), nrow =
          2))
      Y[(nn1 + nn2 + 1):(nn1 + nn2 + nn3), ] <- MASS::mvrnorm(nn3, c(-15, 15), matrix(c(1, 0, 0,
          1), nrow = 2))
      Y[(nh + 1):n, ] <- matrix(runif(p * (n - nh), cc1, 20), ncol = p)
      if (p > 2) {
        Y[1:n, 3:p] <- MASS::mvrnorm(n, rep(0, p - 2), diag(p - 2))
      }
    }
    else {
      Y[1:nn1, ] <- MASS::mvrnorm(nn1, c(5, 5), matrix(c(1, 0, 0, 1), nrow = 2))
      Y[(nn1 + 1):(nn1 + nn2), ] <- MASS::mvrnorm(nn2, c(-15, -15), matrix(c(1, 0, 0, 1), nrow =
          2))
      Y[(nn1 + nn2 + 1):(nn1 + nn2 + nn3), ] <- MASS::mvrnorm(nn3, c(-15, 15), matrix(c(1, 0, 0,
          1), nrow = 2))
      Y[(nh + 1):(nh + nn1), ] <- MASS::mvrnorm(nn1, c(10, -10), matrix(c(1, 0, 0, 1), nrow = 2))
      Y[(nh + nn1 + 1):(nh + nn1 + nn2), ] <- MASS::mvrnorm(nn2,  c(6, -15), matrix(c(1, 0, 0,
          1), nrow = 2))
      Y[(nh + nn1 + nn2 + 1):(nh + nn1 + nn2 + nn3), ] <- MASS::mvrnorm(nn3, c(15, -15),
          matrix(c(1, 0, 0, 1), nrow = 2))
      Y[(2 * nh + 1):n, ] <- matrix(runif((n - 2 * nh) * p, cc1, 20), ncol = p)
      if (p > 2) {
        Y[1:n, 3:p] <- MASS::mvrnorm(n, rep(0, p - 2), diag(p - 2))
      }
    }
  }
  true <- rep(0, n)
  if (G == 3) {
    true[1:nh] <- rep(1:3, c(nn1, nn2, nn3))
  }
  else {
    true[1:(2 * nh)] <- rep(1:6, c(nn1, nn2, nn3, nn1, nn2, nn3))
  }

   return(list(x = Y, true = true))
}
```

2 Additional simulation results

We have considered 100 random realizations of `simula_tclust()` for the parameter combinations discussed in the manuscript, but with configurations of `balance` and `symmetry` that were not included there. The results obtained in all cases are in perfect agreement with those reported in the manuscript. `s=1`, `s=2` and `s=3` refer to the 3 scenarios (`scenario=1`, 2 and 3).

Figure A.1: Target function values when `balance=1` and `symmetry=2`.

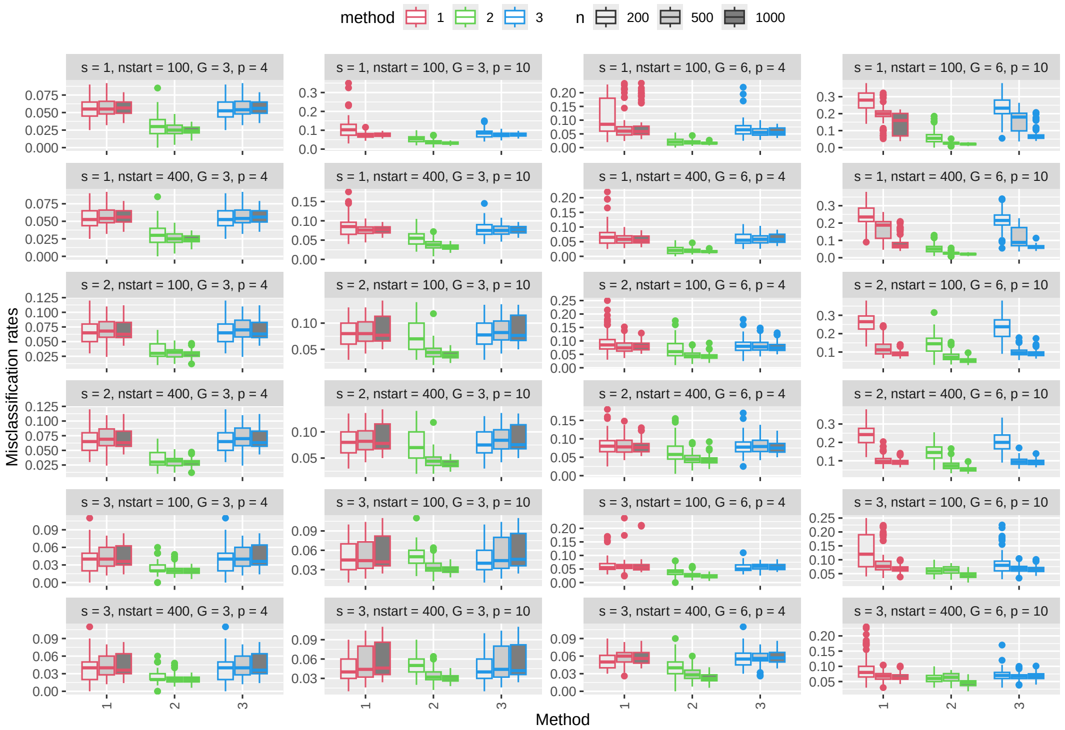


Figure A.2: Misclassification rates when `balance=1` and `symmetry=2`.

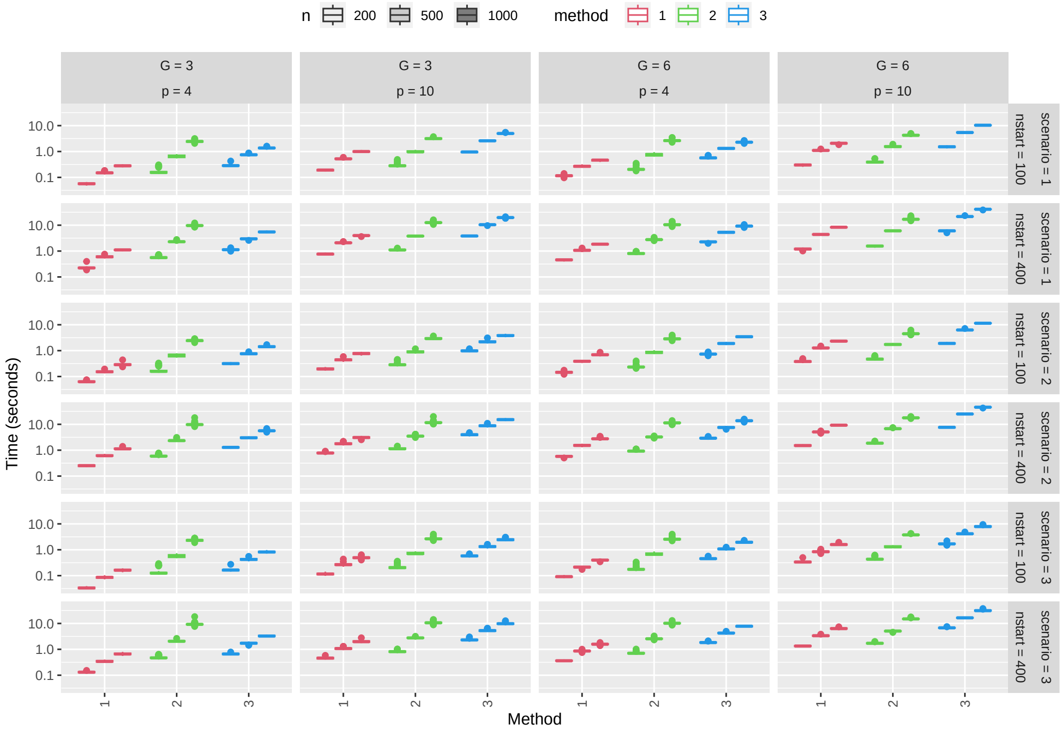


Figure A.3: Computing times in seconds (in logarithmical scale) when `balance=1` and `symmetry=2`.

Figure A.4: Target function values when `balance=2` and `symmetry=1`.

Figure A.5: Misclassification rates when `balance=2` and `symmetry=1`.

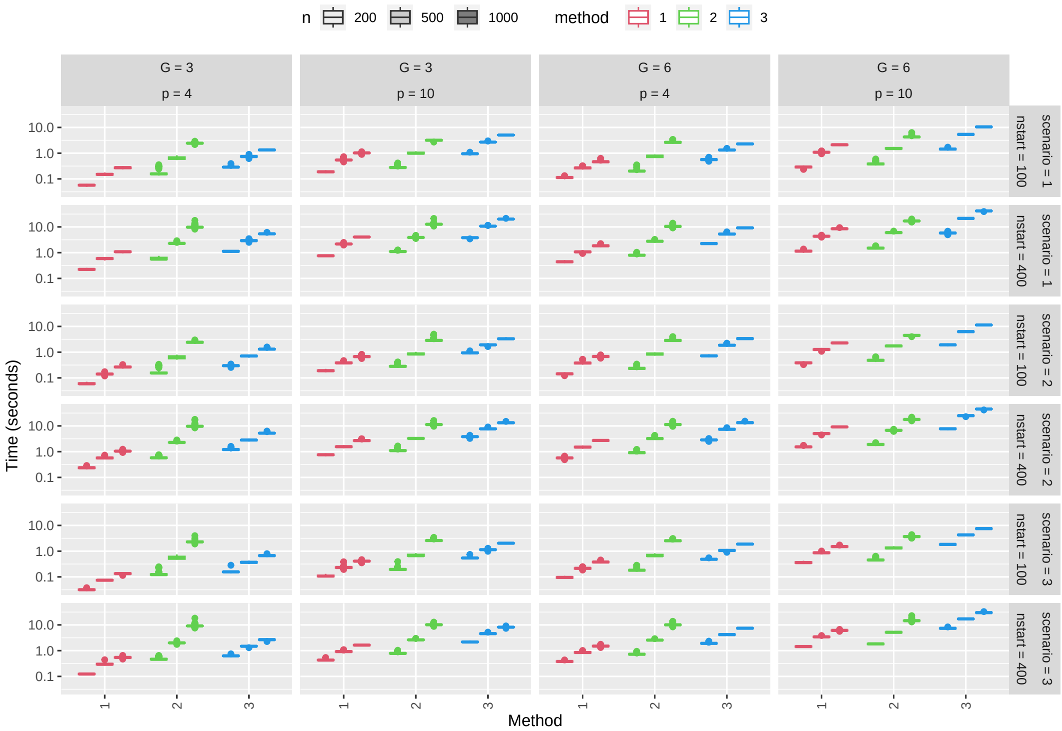


Figure A.6: Computing times in seconds (in logarithmical scale) when `balance=2` and `symmetry=1`.

Figure A.7: Target function values when `balance=2` and `symmetry=1`.

Figure A.8: Misclassification rates when `balance=2` and `symmetry=2`.

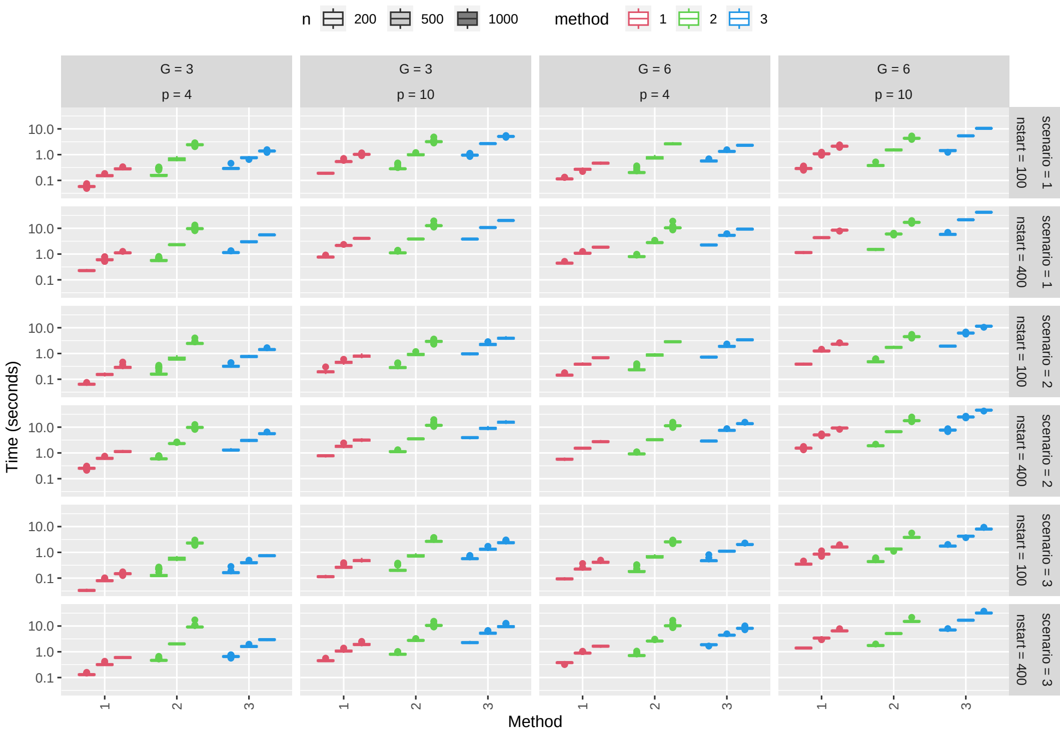


Figure A.9: Computing times in seconds (in logarithmical scale) when `balance=2` and `symmetry=2`.

3 Subsampling and ensemble initialization with $n = 200000$

Results of *Method 1* and *Method 2* for 100 datasets generated by `simula_tclust(n=200000,p,G, scenario, balanced=1, symmetry=1)`. *Method 2* refers to the proposed “Combining subsampling and ensemble initialization” approach with $n_0 = 800$ (notably smaller than $n = 200000$). Results for *Method 3* with $5 \times$ `nstart` are not provided due to the substantial computing times required.

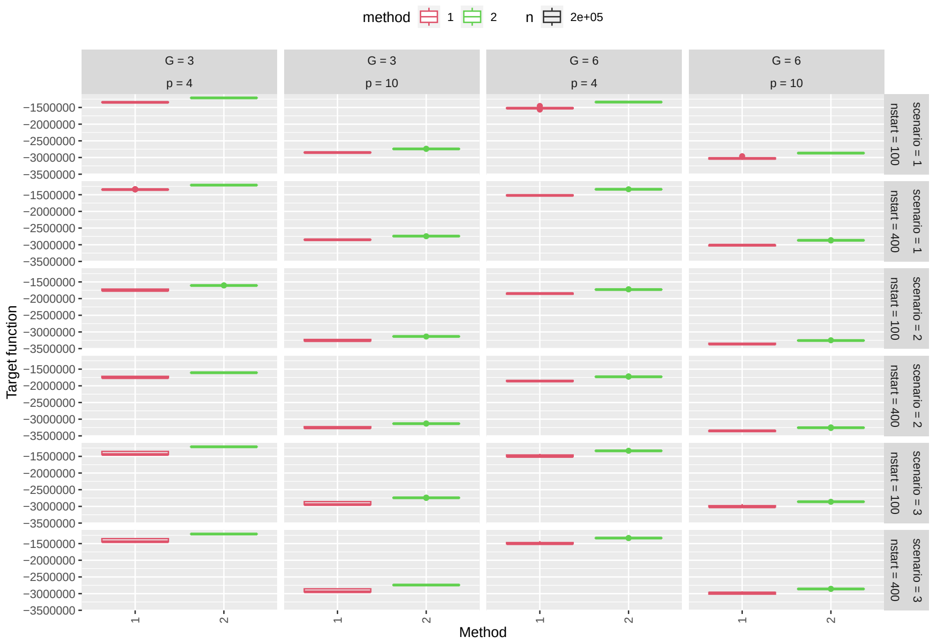


Figure A.10: Target function values when combining subsampling and ensemble initialization with $n = 200000$.

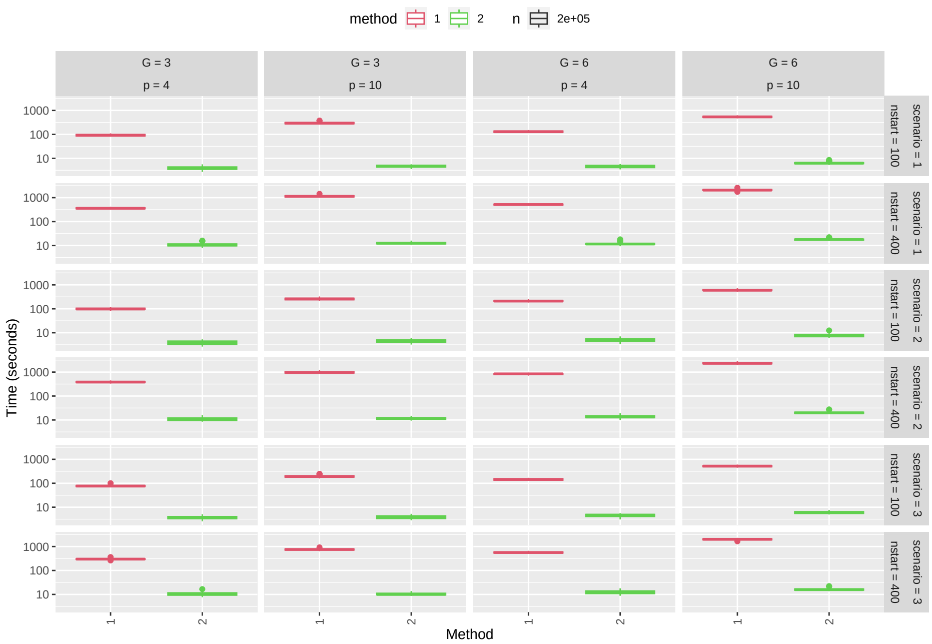


Figure A.11: Computing times (in logarithmic scale) when combining subsampling and ensemble initialization with $n = 200000$.

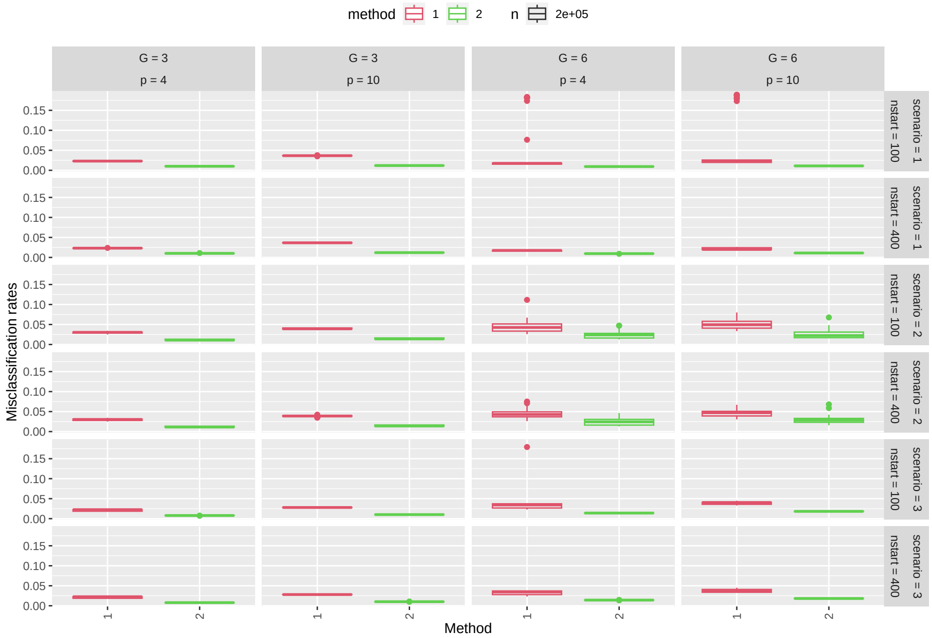
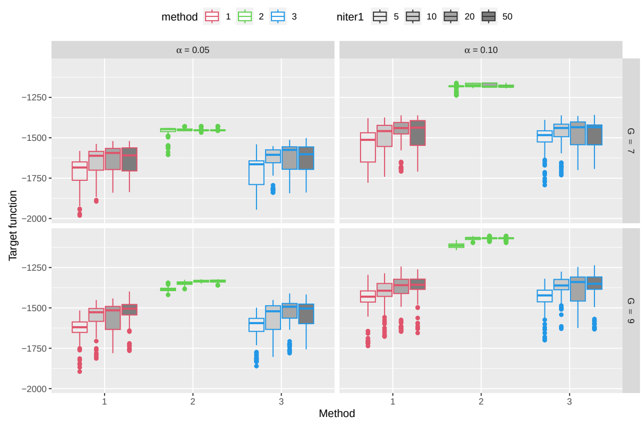
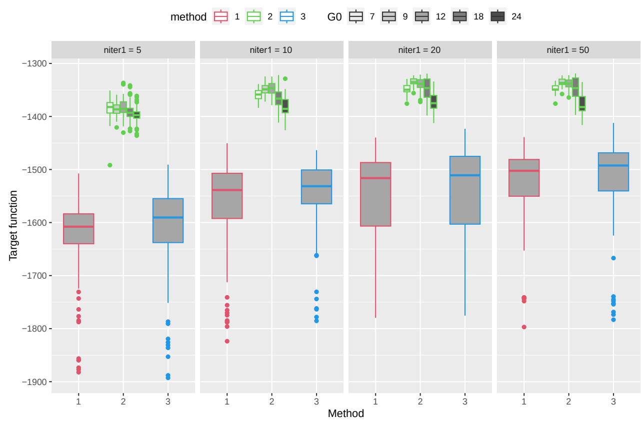


Figure A.12: Misclassification rates when combining subsampling and ensemble initialization with $n = 200000$.

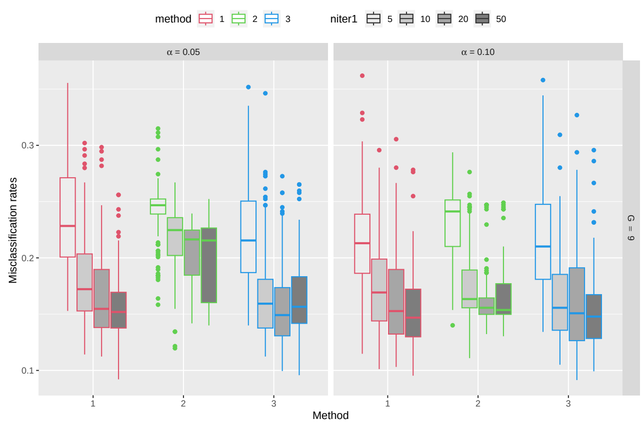
4 Additional results for the "olive oil" dataset



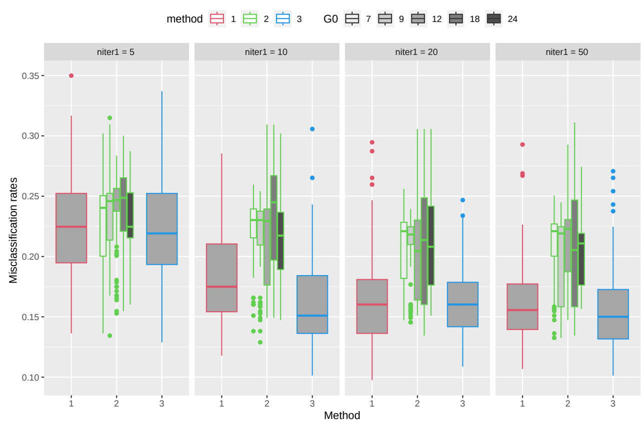
(a)



(b)

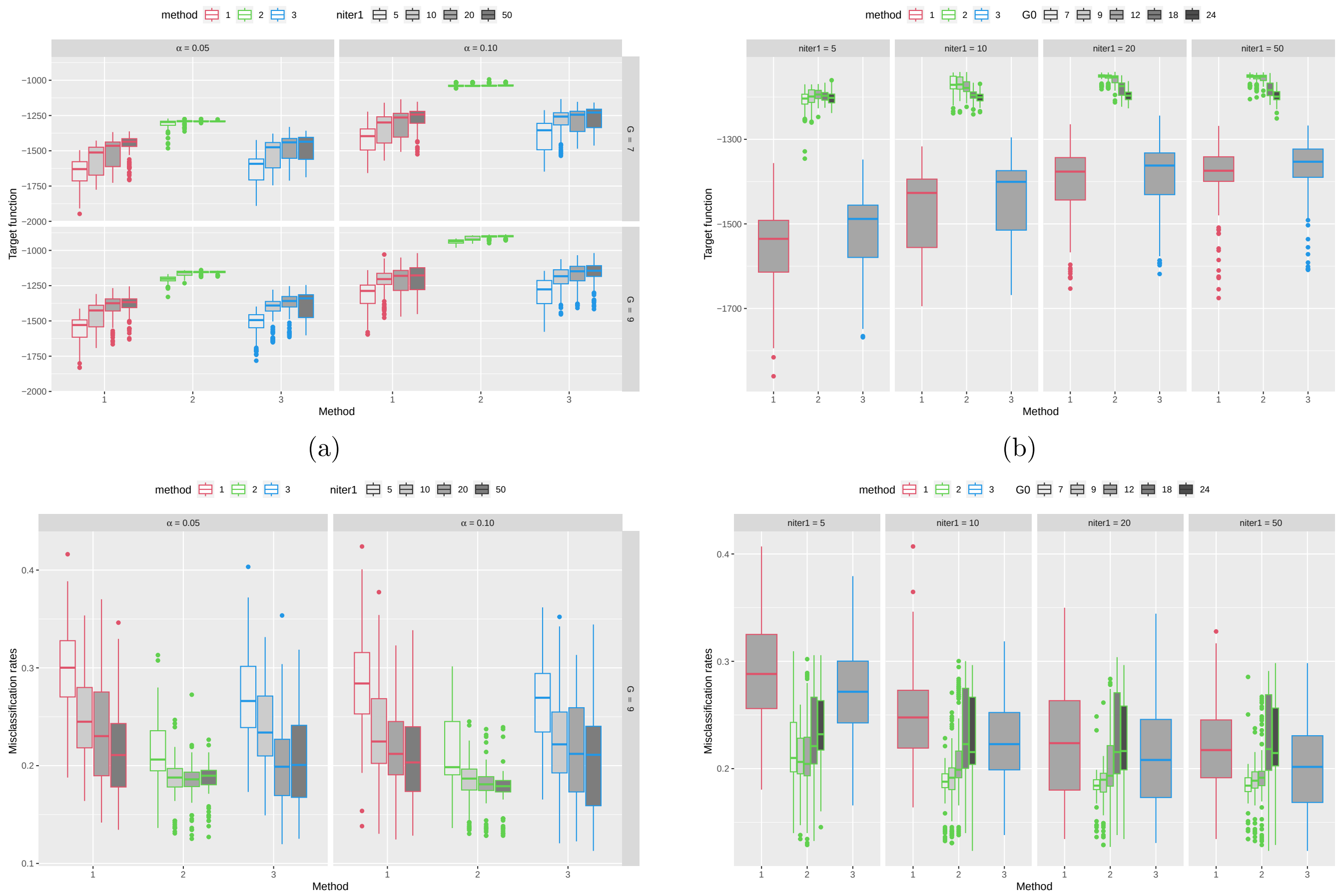


(c)



(d)

Figure A.13: Results of the experiments as in Section 6 of the manuscript when the restriction factor is set to $c = 128$.



(a) (b)

(c) (d)

Figure A.14: Results of the experiments as in Section 6 of the manuscript when the restriction factor is set to $c = 1000$.