Supplement to 'Merged block randomisation: a novel randomisation procedure for small clinical trials'.

Additional simulation results

Abbreviations used in Figure legends: DA = deterministic allocation; PBR = permuted block randomisation; MBR = merged block randomisation; BUD = block urn design; MP = maximal procedure; BSD = big stick design; EBC = Efron's biased coin design; CR = complete randomisation; MTI = maximum tolerated imbalance.

Figure S1: Setting 1 (single centre study). Proportion of suballocations (allocations created by stopping recruitment before the number n is reached) with an imbalance of at least 2, averaged over $N = 1000$ simulation runs.

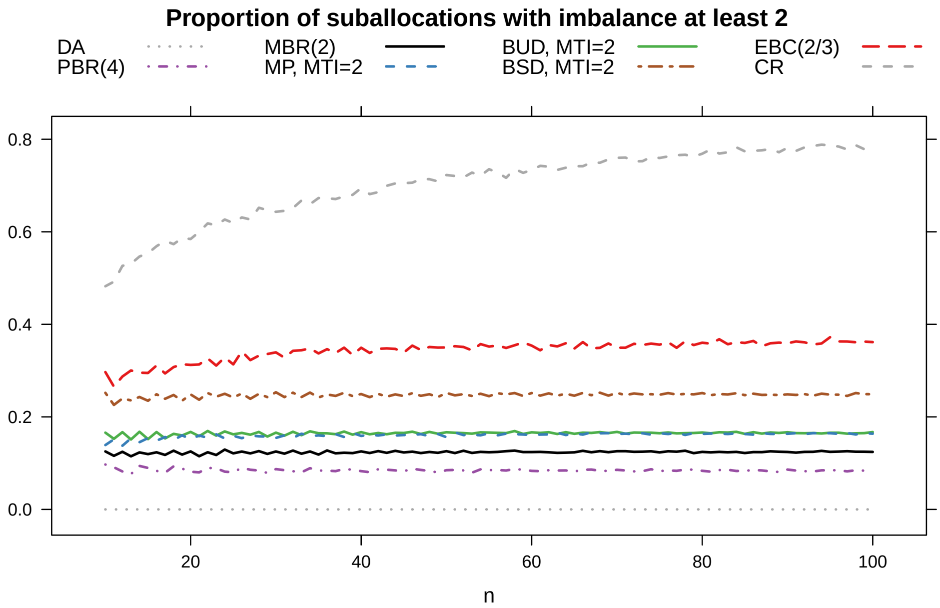


Figure S2: Setting 1 (single centre study). Average correct guess probability of the final allocation over $N = 1000$ simulation runs.

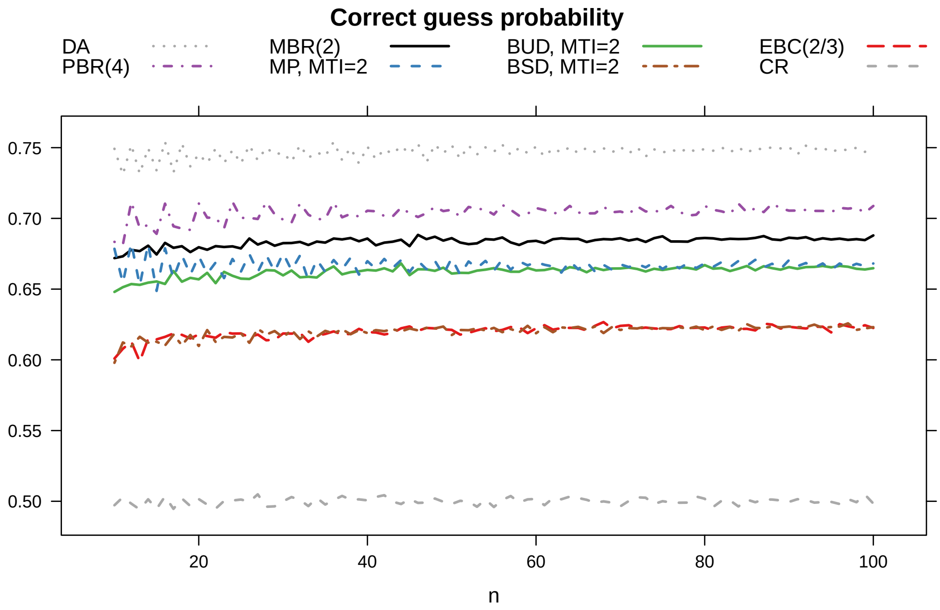


Figure S3: Setting 2 (multicentre study, 10 centres). Average imbalance of the final allocation (combining the ten strata) over $N = 1000$ simulation runs.

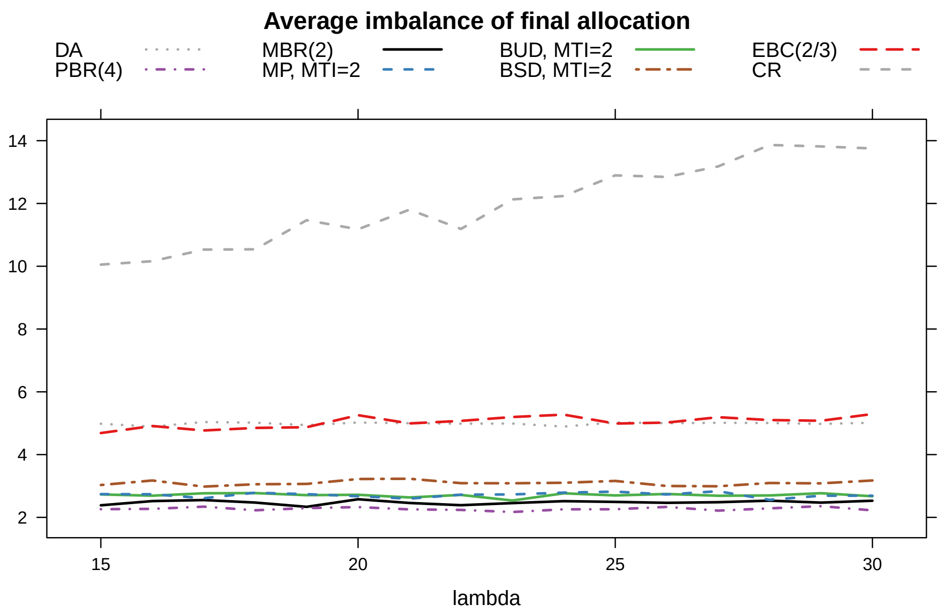
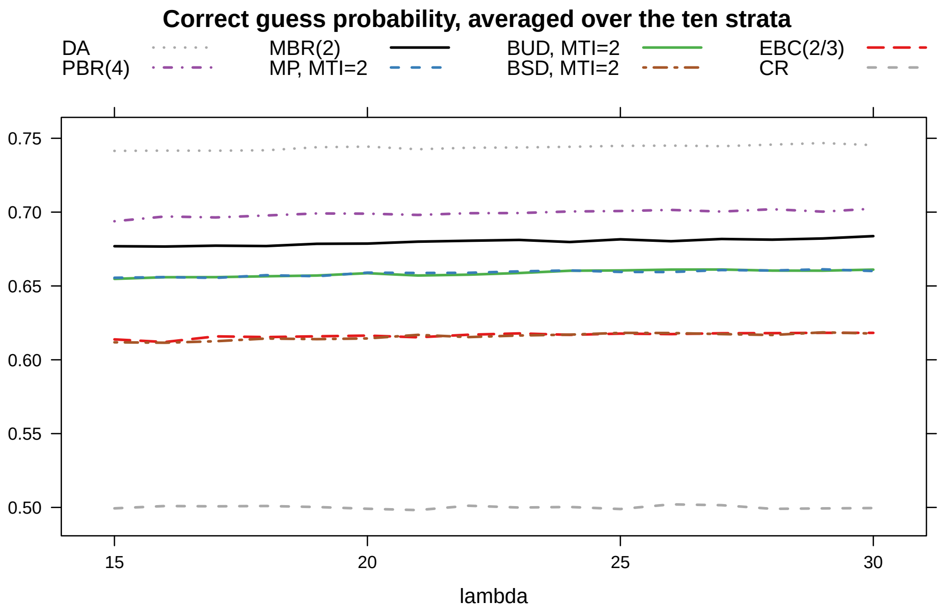


Figure S4: Setting 2 (multicentre study, 10 centres). Average correct guess probability, where the average is taken over the ten correct guess probabilities computed for each of the strata; then averaged over the $N = 1000$ simulation runs.



Pseudo-code

Algorithm 1 Pseudo-code for merged block randomisation

Inputs:
n, the final sample size;
ratio, a vector with the desired allocation ratio, given in integers (e.g. [1 1] for 1:1 allocation).

Initialization
final ← empty vector of length n
counter2 ← 0

Create basis allocations
K ← sum of elements of ratio
basis1 ← PBR(K) of length n, with blocks with the treatment assignment in the given ratio
basis2 ← PBR(K) of length n, with blocks with the treatment assignment in the given ratio
use1or2 ← vector of n fair coin flips, stored as 1's (heads) and 2's (tails)

Merge
for i from 1 to n:
  if use1or2[i] equals 1, **then** final[i] ← basis1[i - counter2]
  else {counter2 ← (counter2 + 1) **and** final[i] ← basis2[counter2]}

R code

```
library(randomizeR)

BlockMergeGen <- function(n, ratio, labels){
  #n = sample size of final allocation
  #ratio = vector with desired allocation ratio, given in integers

  length.blocks <- sum(ratio)
  nr.basis.blocks <- ceiling(n/length.blocks)

  basis1 <- as.numeric(getRandList(genSeq(pbrPar(rep(length.blocks,
  ceiling(n/length.blocks)), K=length(ratio), ratio = ratio, groups = labels))))[1:n]

  basis2 <- as.numeric(getRandList(genSeq(pbrPar(rep(length.blocks,
  ceiling(n/length.blocks)), K=length(ratio), ratio = ratio, groups = labels))))[1:n]

  #now merge
  res <- rep(0, n)
  row1or2 <- sample(c(1, 2), size = n, prob = c(1/2, 1/2), replace = T)

  taken.from.2 <- 0

  for(i in 1:n){

    if(row1or2[i] == 1){res[i] <- basis1[i - taken.from.2]}

    if(row1or2[i] == 2){
      taken.from.2 <- (taken.from.2 + 1)
      res[i] <- basis2[taken.from.2]
    }
  }
  return(res)
}
```