Obtaining Treatment Allocation Sequences by the Maximal Procedure with the MPboost Package

This version was compiled on October 5, 2021

This document provides a short overview of the MPBoost package.

Introduction

The time-honored principle of randomization (Fisher, 1925) has found its way into experimentation in humans through the conduct of clinical trials. An excellent account of the different types of randomization procedures available for assigning treatments in clinical trials may be found in Rosenberger and Lachin (2016). As explained there, although applying the procedure of *complete randomization* could seem an obvious way to proceed, it suffers from an important drawback. Complete randomization can give rise to unbalanced treatment assignments, resulting in a loss of power of the tests applied. As a way of forcing assignment balance, *restricted randomization* procedures have been developed. The feature in common to all restricted randomization procedures is the probabilistic dependence of any assignments (excepting for the first one) on previous assignments. As a matter of fact, nowadays the majority of clinical trials resort to one of the different procedures of restricted randomization available to ensure balance in the treatment assignments. However, excessively restrictive procedures are exposed to selection bias. This happens, e.g., in unmasked trials with a *permuted block* design of fixed block size. As an extreme example, consider the case of a two-armed clinical trial with treatment ratio 1:1, in which the first M patients of a block of size $2M$ are allocated to treatment 1; then, the last M allocations must be to treatment 2 and are entirely predictable by the researcher.

The *maximal procedure* (MP) of allocation was devised by Berger *et al.* (2003) as an extension of permuted block procedures in which the feasible sequences are those with imbalance not larger than a *maximum tolerated imbalance* (MTI), all of them being equiprobable. They proved that MP has less potential for selection bias than the *randomized block* procedure. Also, it compares favorably with *variable block* procedures under some likely scenarios.

MPBoost

Overview of the package. Salama *et al.* (2008) proposed an efficient algorithm to generate allocation sequences by the maximal procedure of Berger *et al.* (2003). `MPBoost` is an `R` package that implements the algorithm of Salama *et al.* (2008). This algorithm proceeds through the construction of a directed graph, and so does its implementation in `MPBoost`, using to that end the functionality provided by the `Boost Graph Library` (BGL). BGL is itself a part of `Boost C++ Libraries` (Boost Community, 2019). Although the recommended reference for BGL is the updated documentation in Boost Community (2019), the reader unacquainted with BGL can find a useful guide in Siek *et al.* (2002).

MP may generate a huge reference set of feasible sequences (Berger *et al.*, 2003; Salama *et al.*, 2008). In my implementation, in order to ensure that the probabilistic structure of the procedure is correctly reproduced, the number of sequences is computed by using multiprecision integer arithmetic. I have opted for the `Boost Multiprecision Library`, also a part of `Boost C++ Libraries` (Boost Community, 2019), taking more into account the easiness of its integration with `R` than any efficiency issues. Anyway, in spite of the huge computational burden introduced by this approach, sequences of length greater than any practical value (e.g., in the order of thousands) are very efficiently computed.

In the package, interfacing between `R` and `C++` code has been addressed with the aid of the `Rcpp` package (Eddelbuettel and François, 2011).

Using the package. The package consist of only one `R` function: `mpboost()`. Its arguments are `N1`, `N2`, and `MTI`. The arguments `N1` and `N2` specify the numbers allocated to the treatments 1 and 2, respectively. The MTI is set through the `MTI` argument, whose default value is 2. The value returned by `mpboost()` is an integer vector of N1 1's and N2 2', representing the sequence of treatments 1 and 2 allocated by the realization of the MP. The following code illustrates a typical call:

```
library(MPBoost)
mpboost(N1 = 6, N2 = 6)
```

```
#   [1] 1 1 2 1 2 2 1 2 1 2 2 1
```

In the next example the optional MTI is set to a value different from the default:

```
mpboost(N1 = 6, N2 = 6, MTI = 3)
```

```
#   [1] 2 2 2 1 1 2 1 1 2 2 1 1
```

Allocating sequences with treatment ratios different to 1:1 is also possible, as the next code illustrates:

```
mpboost(N1 = 6, N2 = 12)
```

```
#   [1] 2 1 1 2 2 2 2 1 1 2 2 2 1 1 2 2 2
```

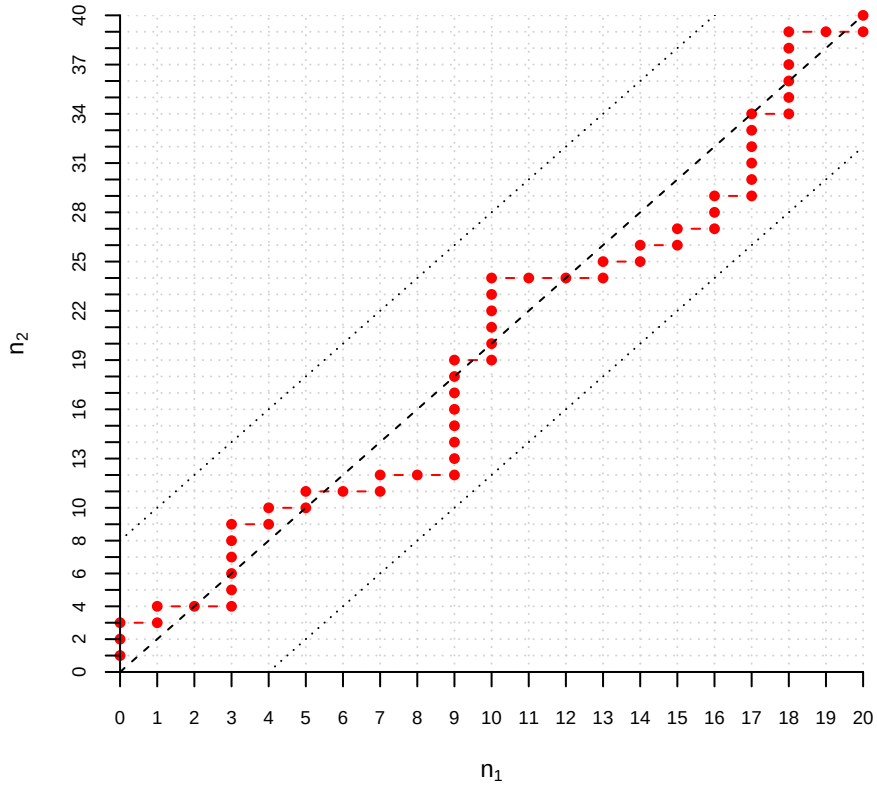
Of course, in actual clinical trials, longer sequences should be needed:

```
set.seed(1) ## Only needed for reproductibility
x1 <- mpboost(N1 = 20, N2 = 40, MTI = 4)
x1
```

```
#   [1] 2 2 2 1 2 1 1 2 2 2 2 2 1 2 1 2 1 1 2 1 1 2 2 2 2 2 2 1 2 2 2 2 2 1 1 1 2
#  [39] 1 2 1 2 1 2 2 1 2 2 2 2 2 1 2 2 2 2 2 1 1 2
```

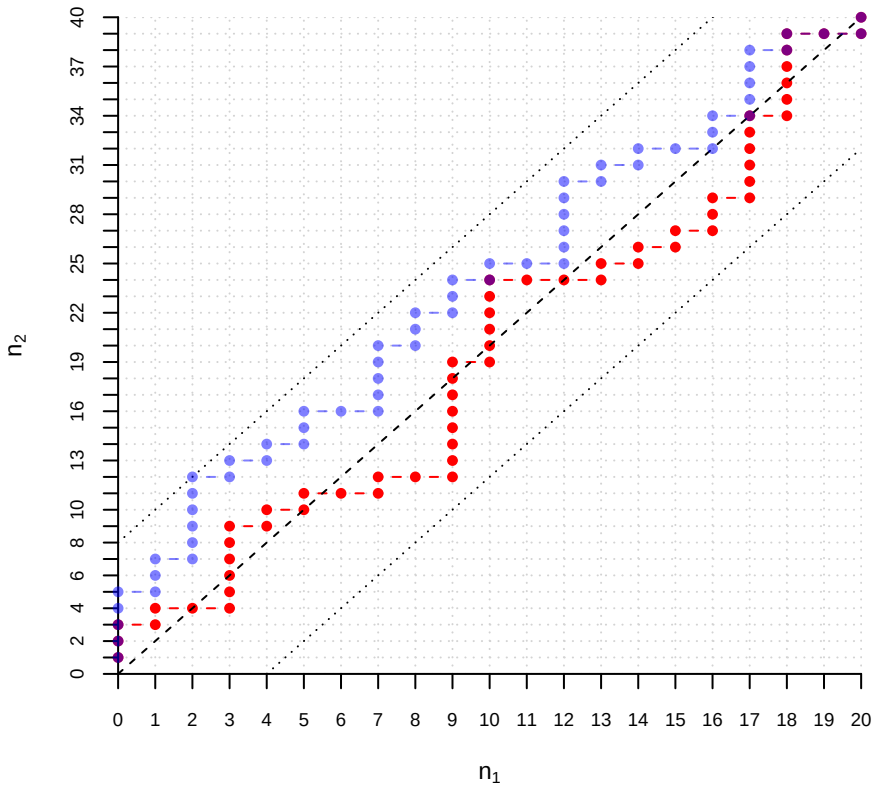
Running the following code, a plot illustrating the dynamics of allocation and the relationship with the constraints set by the MTI is created:

```
lx <- sum(x1 == 1)
ly <- sum(x1 == 2)
ratio <- lx/ly
mti <- 4
plot(cumsum(x1 == 1), cumsum(x1 == 2), xlim = c(0, lx), ylim = c(0, ly),
   xlab = expression(n[1]), ylab = expression(n[2]), lab = c(lx, ly, 7),
   type = "b", pch = 16, panel.first = grid(), col = "red", bty = "n",
   xaxs = "i", yaxs = "i", xpd = TRUE, cex.axis = 0.8)
abline(-mti/ratio, 1/ratio, lty = 3)
abline(mti/ratio, 1/ratio, lty = 3)
abline(0, 1/ratio, lty = 2)
```



The code below redraws the previous plot and superimposes another sequence on it (note that in the new sequence the MTI is reached once):

```r
set.seed(11) ## Only needed for reproductibility
x2 <- mpboost(N1 = 20, N2 = 40, MTI = 4)
plot(cumsum(x1 == 1), cumsum(x1 == 2), xlim = c(0, lx), ylim = c(0, ly),
   xlab = expression(n[1]), ylab = expression(n[2]), lab = c(lx, ly, 7),
   type = "b", pch = 16, panel.first = grid(), col = "red", bty = "n",
   xaxs = "i", yaxs = "i", xpd = TRUE, cex.axis = 0.8)
abline(-mti/ratio, 1/ratio, lty = 3)
abline(mti/ratio, 1/ratio, lty = 3)
abline(0, 1/ratio, lty = 2)
lines(cumsum(x2 == 1), cumsum(x2 == 2), type = "b", pch = 16,
   col = rgb(0, 0, 1, alpha = 0.5), xpd = TRUE)
```



Other implementations. To my knowledge, the package `randomizeR` (Uschner *et al.*, 2018) is the only additional package on CRAN that implements the MP (as of October 1, 2019). Concerning randomization methods, `randomizeR` has a much broader scope than `MPBoost`, as the ample variety of procedures implemented by the package demonstrates. As for the programming style, `randomizeR` uses `S4` classes but not compiled code (specifically, it makes no use of either `C++` code or public `C++` libraries). Related to the latter feature, the implementation of MP in `MPBoost` seems to be more computationally efficient than that of `randomizeR`. This advantage would show up when long sequences are generated or in simulation studies.

References

Berger VW, Ivanova A, Knoll MD (2003). "Minimizing predictability while retaining balance through the use of less restrictive randomization procedures." *Statistics in Medicine*, **22**, 3017–3028. doi:10.1002/sim.1538.

Boost Community (2019). "Boost C++ Libraries." Accessed: 2019-09-30, URL http://www.boost.org/.

Eddelbuettel D, François R (2011). "Rcpp: Seamless R and C++ Integration." *Journal of Statistical Software*, **40**(8), 1–18. doi:10.18637/jss.v040.i08. URL http://www.jstatsoft.org/v40/i08/.

Fisher RA (1925). *Statistical Methods for Research Workers*. Oliver & Boyd, UK.

Rosenberger WF, Lachin JM (2016). *Randomization in Clinical Trials: Theory and Practice*. 2 edition. John Wiley & Sons, Hoboken, NJ, USA. ISBN 978-1-118-74224-2.

Salama I, Ivanova A, Qaqish B (2008). "Efficient generation of constrained block allocation sequences." *Statistics in Medicine*, **27**, 1421–1428. doi:10.1002/sim.3014.

Siek JG, Lee LQ, Lumsdaine A (2002). *The Boost Graph Library: User Guide and Reference Manual*. Addison-Wesley, Boston, MA, USA. ISBN 0-201-72914-8.

Uschner D, Schindler D, Hilgers RD, Heussen N (2018). "randomizeR: An R package for the assessment and implementation of randomization in clinical trials." *Journal of Statistical Software*, **85**(8), 1–22. doi:10.18637/jss.v085.i08.