Appendix

To utilize this appendix, the statistical program R version 3.6.0 (R Core Team (2019). R: A language and environment for statistical computing, R Foundation for Statistical Computing, Vienna, Austria. R-project.org/) is needed. It is also necessary to understand the installation of its package, data path specification, input and execution of the command, and so on.

There are a few packages that make it easier to do randomization. In this appendix, we use "randomizeR" package [1]. This package contains functions and commands for creating various randomization sequences.

The following command invokes the randomizeR package.

```
> library(randomizeR)
```

Let's try some of the randomization methods introduced in this article.

1. Simple randomization

The function for simple randomization is crPar, which defines and includes the basic elements needed for a random sequence. The genSeq function then generates a randomization sequence. Let's look at simple randomization for a study in which the total number of subjects is 1000 (N) and the number of groups is 3 (K).

First, specify rand with basic information for full randomization. If you open this rand, you can check the stored information.

```
> N ← 1000
> K ← 3
> (rand ← crPar(N, K))

Object of class "crPar"
design = CR
N = 500
K = 3
groups = A B C
```

Next, run the genSeq function with rand to generate a random sequence, and specify it as crs.

```
> (crs ← genSeq(rand))
Object of class "rCrSeq"

design = CR
seed = 476000002
N = 500
K = 3
groups = A B C

The sequence M:

1 B C B B B C B C A B ...
```

Here you can see only some of the entire random sequence, so to see the whole,

```
> getRandList(crs)
```

Note that a new random sequence is generated each time you run it. If you want to see the random sequence already generated, you need to keep the seed[1)] value. The seed generated here is 476000002. If you generate crs.2 with the seed, crs.2 can be equal to crs.

```
> (crs.2 ← genSeq(rand, 476000002))
> getRandList(crs.2)
```

The generated random sequence can be saved as a 'simpleRS.csv' file with the following command.

```
> saveRand(crs, file = "simpleRS.csv")
```

2. Block randomization

This randomization method is performed using one block size. For example, suppose there is a study in which a total of 90 subjects are allocated to three groups (K = 3), each with 30 subjects. If six is selected as the block size (bc), 15 iterations (R = 15) are needed for allocation of 90 subjects (if the block size is 9, you will need 10 iterations). First, basic information is assigned to rand by using pbrPar, a function for block randomization. Next, the allocation order generated by the genSeq function with 'rand' is stored in brs.

```
> bc ← 90
> K ← 3
> R ← 15
> (rand ← pbrPar(bc, K))
> (brs ← genSeq(rand, R))

Object of class "rPbrSeq"

design = PBR(6)
seed = 408740403
N = 6
K = 3
groups = A B C
bc = 6

The first 3 of 15 sequences of M:

1 C C B A B A
2 B C C A A B
3 C B A C A B
...

> getRandList(brs)
```

To save the generated random sequence (brs) as a csv file

```
> saveRand(brs, file = "blockRS.csv")
```

A seed value is needed to regenerate and confirm the same random sequence. If you apply the seed value given during brs genera-

[1)]Whenever a new random sequence is generated, the seed value will change.

tion to generate brs.2, you can see the same random sequence as brs.

```
> (brs.2 ← genSeq(rand, 408740403))
> getRandList(brs.2)
```

3. Randomized block randomization

The default function is rpbrPar. The researcher determines the total number of participants (N = 80) and the randomly chosen block size (rb, size of 3, 6, 9). The basic function repeats the selection of any block size (one of 3, 6, or 9) and the allocation of subjects within the selected block until all subjects are assigned.

If the last block size is larger than the remaining number of subjects, the last block is not filled. If the last block should be filled, set filledBlock = FALSE. Otherwise, set filledBlock = TRUE. The random sequence is generated and stored in rbrs. BlockConst shows that blocks of various sizes are used.

```
> N ← 60
> rb ← c(3, 6, 9)
> K ← 3

> rand ← rpbrPar(N, rb, K, ratio = rep(1, K), groups = LETTERS[1:K], filledBlock = FALSE)
> (rbrs ← genSeq(rand))

Object of class "rRpbrSeq"

design = RPBR(3,6,9)
rb = 3 6 9
filledBlock = FALSE
seed = 939989023
N = 60
K = 3
ratio = 1 1 1
groups = A B C

RandomizationSeqs BlockConst
B C A C B C B A ... 3 9 9 6 3 ...
```

You can view or save the random sequence generated by the following command.

```
> getRandList(rbrs)
> saveRand(rbrs, file = "RBRS.csv")
```

4. Random allocation rule

Te jar model is often used. Put as many balls as the number of subjects (the color of the ball varies according to the group) in the jar according to the allocation ratio. Whenever the subject is assigned, take out the ball and confirm it, and do not put the ball back into the jar (random sampling without replacement). This allocation is repeated on a block-by-block basis. The following is the order to apply the random allocation rule to compare the three groups (K) with the total number of subjects (N) of 60.

```
> N ← 60
> K ← 3
> rand ← rarPar(N, K)
> (rar ← genSeq(rand))

Object of class "rRarSeq"

design = RAR
seed = 144561
N = 60
K = 3
groups = A B C

The sequence M:

1 A B B C A B A B C C ...
```

You can view or save the random sequence generated by the following command.

```
> getRandList(rar)
> saveRand(rar, file = "RAR.csv")
```

5. Stratified randomization

Once the prognostic factors and strata have been determined, randomization for each stratum can be performed using the methods previously described. However, random sequences are required as many as the number of strata. In the previous example, the prognostic factors were the recruitment hospitals (site 1, 2), sex (male, female), and age band (under 20 years old, 20-64 years, 65 years old or older). Therefore, 12 random sequences are required.

Reference

1. Uschner D, Schindler D, Hilgers RD, Heussen N. randomizeR: an R package for the assessment and implementation of randomization in clinical trials. J Stat Softw 2018; 85: 1-22.