Introduction to clinical trials
Lab 3: Randomization schemes

1 Introduction

Randomization is a method used in clinical trials in order to make sure that each subject will be assigned to the experimental group or to the control group randomly. Thus, treatment assignment is a matter of chance, not depending on the patient's characteristics.

There are three main reasons that motivate the use of randomization in clinical trials. First, clinical outcomes often have large variability compared to the magnitude of differences or effects that treatments are likely to produce. A good study, including randomization, can help us control this variability. The second motive is a selection-bias issue. In many cases, clinicians may assign patients with certain characteristics to a particular treatment with higher probability. For example, if a more severely ill patient is more likely to receive treatment, receiving treatment is associated with disease severity, which could erroneously show that treatment is the cause of the severity of the disease. In this case, treatment effect is confounded with patient's prognostic factors. Third, randomization can reduce patients heterogeneity, ensuring that the subject groups will be as homogeneous as possible.

Although statistical models can adjust the treatment differences for confounding, there are some limitations in practice. First, statistical models are able to prevent confounding of the treatment outcome only by variables measured (such as age, sex or other problem-specific covariates). Thus, confounding due to unmeasured factors (e.g. genetic or environmental effects) can't be eliminated. Instead, randomization ensures balance of the unmeasured factors between the two groups. There is also something else that people tend to ignore: a statistical model can successfully prevent confounding only if the model for the relationship between the outcome and confounders is approximately correct and other underlying assumptions of the model are met.

2 Simple Randomization

This method ensures that the number of patients won't be different between the two groups in the long run, but in infinite samples it may yield severe imbalances.

If we try to randomize by simply flipping a coin, the size of the two treatment groups will not, in general, be equal. Letting N be the total sample size and N_A be the sample size in group A, and recalling that the N_A follows the binomial distribution, $B(N, 1/2)$, the probability of getting equal sample sizes is

$$\Pr(N_A = N/2) = \binom{N}{N/2}\frac{1}{2^N}.$$

For example, when $N = 100$, the chance of the randomization yielding exactly 50 subjects per group is only about 8%. In other words, there is only 8% probability that we will get two balanced groups of 50 patients if we randomize in this manner. In R, you can use the dbinom function to get the probability mass function of the binomial distribution.

```
##############################
### Randomization schemes ###
##############################

# (ii)-(a) We expect to randomize 100 patients
# What's the probability of getting equal sample size?
N = 100
dbinom(x = N/2, size = N,p = 0.5)

## [1] 0.07958924
```

Flipping a coin is a random experiment in the sense that the probabilities of coming up heads and tails are both equal to 0.5. We can simulate this experiment using random number generation. We're not going to go into detail on simulation, but the whole idea is based on the simulation of uniform random numbers on the interval $(0, 1)$. The uniform distribution assigns the same probability to all points within its support. Thus, to simulate the occurrence of an event that is happening about 80% percent of the times, we can simulate a random variable u and if it's less than 0.8, we assume that the event has occurred. For our purposes, the event is the assignment of a patient to a particular treatment (say treatment A, for example). We want the probability of this event to be 0.5. Using R.

2.1 Simulation experiment

```
# (ii)-(b) Simulate an example of simple randomization
p = 0.5
N = 100
data = data.frame(id = paste("id",1:N,sep = ""), trt = NA)
head(data)

##    id trt
## 1 id1  NA
## 2 id2  NA
## 3 id3  NA
## 4 id4  NA
## 5 id5  NA
## 6 id6  NA

 # Set the seed if you would like to reproduce your results
set.seed(125)
```

```
# Make assignments
data$trt[u<0.5] = "A"
data$trt[u>0.5] = "B"
data[1:10,]
```

```
# Simulate 100 uniform random variable
u = runif(N)
hist(u, main="")
```

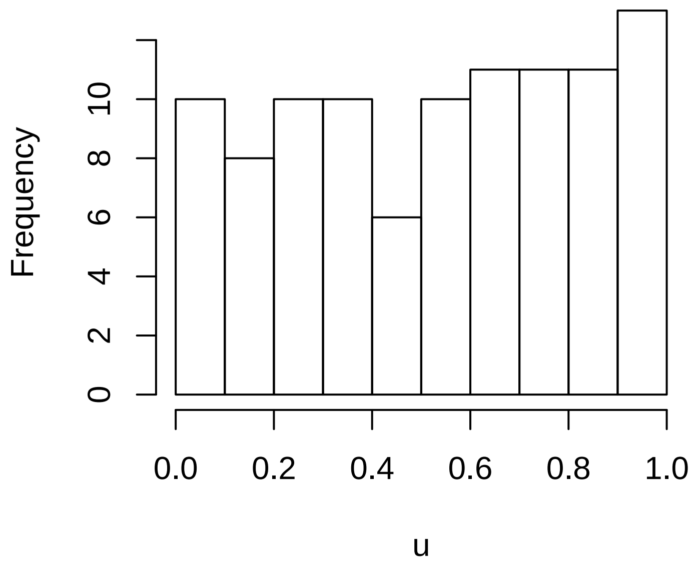

Figure 1: Histogram of uniform random numbers

```
##      id trt
## 1   id1   B
## 2   id2   A
## 3   id3   A
## 4   id4   A
## 5   id5   B
## 6   id6   B
## 7   id7   B
## 8   id8   A
## 9   id9   B
## 10 id10   B
```

```
table(data$trt)
```

```
##
##  A  B
## 44 56
```

```
table(data$trt)/N
```

```
##
```

```
##    A    B
## 0.44 0.56
```

Thus, in our experiment we got 56 patients randomized on treatment B and 44 randomized patients on treatment A.

Letting $p = 0.5$ be the probability of making an assignment to treatment A or B, by standard probability theory, the expected number of assignments to each of two groups is

$$\mathrm{E}(N_A) = \mathrm{E}(N_A) = Np$$

and the corresponding variance

$$\mathrm{Var}(N_A) = \mathrm{Var}(N_A) = Np(1-p).$$

Thus, when $N = 100$ and $p = 0.5$, using normal approximation to the binomial distribution, an approximate 95% confidence bound on the number of assignments to treatment A is $N_A \pm 1.96 \times \sqrt{25} \approx N_A \pm 10$. Thus we can expect more than a 60/40 imbalance in favor of A (or B) 5% of the time using simple randomization. If we repeat the (simple) randomization 1000 times, we will expect to observe such an imbalance in about 50 replications.

2.2 Further simulations of an SR experiment

We repeat the same experiment 1000 times. For each replication, we make simple randomization of $N = 100$ subjects to treatment A and treatment B and store the actual numbers of patients on treatment A and B in a data frame. Then we calculate the average number of patients in each group across the 1000 replications, which is expected to be 50. Since an approximate CI for the number of patients on treatment A is $N_A \pm 10$, we expect to observe imbalance greater than 60/40 in about 50 replications.

```r
# (ii)-c: Repeating simple randomization 1000 times
K = 1000
set.seed(2342)

results = data.frame(replication = 1:K, NumberA = NA, NumberB = NA)

for (i in 1:K)
 {
   data = data.frame(id = paste("id",1:N,sep = ""), trt = NA)
   u = runif(N)

   # Make assignments
   data$trt[u<0.5] = "A"
   data$trt[u>0.5] = "B"

   results[i,"NumberA"] = table(data$trt)["A"]
   results[i,"NumberB"] = table(data$trt)["B"]
 }
head(results)
```

```
##   replication NumberA NumberB
## 1           1      39      61
## 2           2      48      52
## 3           3      47      53
## 4           4      53      47
## 5           5      44      56
## 6           6      50      50

# Number of replications yielding imbalance greater than 60/40
table(results[,2] > 60 | results[,2] < 40)/K

##
## FALSE  TRUE
## 0.958 0.042
```

We can also look at the means and variances

```
# See means and variances
# Note the equal number on each treatment in the long run
apply(results[,2:3],2,mean)

## NumberA NumberB
##  49.887  50.113

apply(results[,2:3],2,var)

##  NumberA  NumberB
## 28.05228 28.05228
```

In the following figure we see that the long-term probability converges indeed to 50% but this happens only when the sample size is very large The mean number of assignments to treatment A is approximately 50, thus the simple randomization reaches equal group allocation in the long run. Also, we observe a number of patients on treatment A greater than 60 or less than 40 in 4.2% of the replications, confirming that an approximate 95% confidence bound on the number of assignments to treatment A is $(40, 60)$.

3 Randomization in blocks

If there are two treatments, A and B, and the blocks are of size four, there are $\binom{6}{2} = 6$ different permutations of the assignments within blocks. The function `choose` returns the binomial coefficient $\binom{n}{k}$, whereas the function `perm` enumerates all permutations of a vector.

```
# (iii) Randomization in blocks
# (a) block size 4 and two treatments A and B
# Number of permutations
choose(4,2)

## [1] 6
```

```r
# Notice that Na converges to 50
plot(cumsum(results[,2])/1:K,type = "l",xlab = "Replication index",
      ylab = "Mean number of patients on treatment A")
```

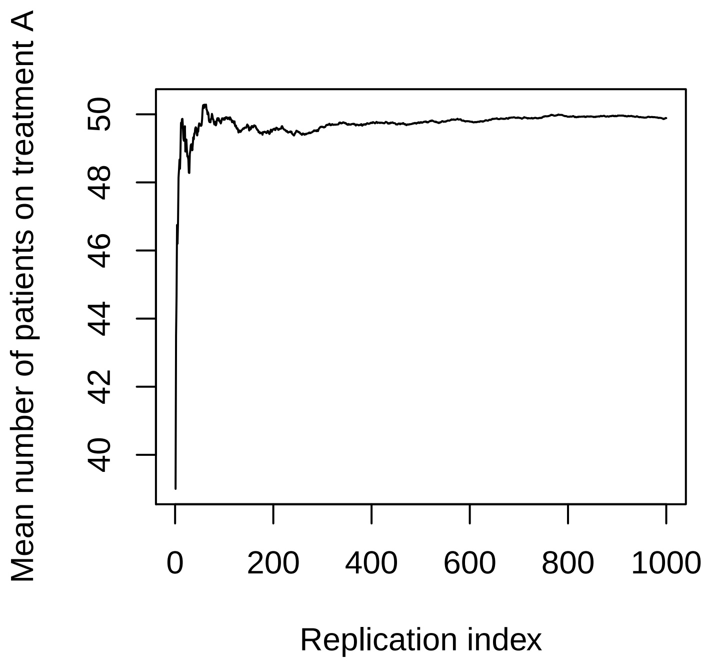

Figure 2: Long-term proportion of $N = 100$ subjects assigned to two treatments

Randomization in blocks ensures balance at the end of the blocks. However, if the procedure stops in the middle of a block in which the first two assignments are A, the imbalance will be two extra A compared to B. This reveals that the maximum imbalance is one-half of the block size.

To implement this with `R`, we first import the library `blockrand`. The function `blockrand` carries out block randomization lists with randomly chosen block sizes. Note that we use a seed to get reproducible results.

```r
# (c) Randomize 100 subjects using random block sizes
library(blockrand)

set.seed(1322)
block_rand = blockrand(n = 100, num.levels = 2, levels = c("A","B"),
                       block.sizes = 1:4)
block_rand[block_rand$block.id %in% 1:2,]

##    id block.id block.size treatment
## 1   1        1          4         A
## 2   2        1          4         A
## 3   3        1          4         B
## 4   4        1          4         B
## 5   5        2          6         B
## 6   6        2          6         B
## 7   7        2          6         A
## 8   8        2          6         A
## 9   9        2          6         A
## 10  10       2          6         B
```

We specified the number of subjects to be randomized (`n = 100`), the number of treatment arms (`num.levels = 2`) and the treatment labels (`levels=c("A","B")`). The function returns a data frame with subject and block identifiers, *id* and *block.id*, respectively. Also, there is a variable denoting the size of each block. Note that the number of subjects randomized (i.e. the number of rows of `block_rand`) can be greater than 100. Thus, since the procedure stops when 100 subjects have been randomized, we may end up with unequal sample sizes in the two groups.

```
# Actual number of randomizations made
nrow(block_rand)

## [1] 102

# Number of patients in each group
table(block_rand$treatment[1:100])

##
##  A  B
## 51 49
```

We got $n_1 = 51$ patients in group A and $n_2 = 49$ patients in group B, which is very close to a balanced group allocation.

Since some assignments within each block are not random (i.e. determined from the previous assignments), knowledge of block size will get future assignments predictable. Thus, the block size should not be revealed to the clinical investigators. Instead, random block sizes can make the randomization list appear more random and prevent discovery of future assignments.

4 Blocking within strata

Since b assignments are made in each block, the maximum imbalance is b in each block. Thus, the size of the maximum imbalance resulting from k strata is $b \times k$. However, we would have to be extremely unlucky for the actual imbalance to be at the maximum when the study ends.

We carry out randomization in blocks for men and women separately and keep 55 men and 45 women.

```
# (iv): Blocking within strata
# (b) Suppose we stratify by sex
set.seed(2435)
```

```r
male=blockrand(n=100,id.prefix='M', block.prefix='M',stratum='Male')
female=blockrand(n=100,id.prefix='F', block.prefix='F',stratum='Female')

# 55 men and 45 women at the end of the study
male = male[1:55,]
female = female[1:45,]

# Combine into one dataset
mystudy = rbind(male,female)

# Number of subject in the two groups
table(mystudy$treatment)

##
##  A  B
## 50 50

# Two-way contingency table
x = table(mystudy$treatment,mystudy$stratum)
x

##
##     Female Male
##   A     22   28
##   B     23   27

# Chi-square test
chisq.test(x,simulate.p.value = T,B = 10000)

##
##  Pearson's Chi-squared test with simulated p-value (based on 10000
##  replicates)
##
## data:  x
## X-squared = 0.040404, df = NA, p-value = 1
```

In this experiment, we got 50 patients randomized to A and 50 patients to B (a perfect balance). Also, since we used randomization in blocks separately in men and women, the proportion of patients assigned to A is almost equal to the proportion of patients assigned to B in both men and women. Thus, as expected, the chi-squared test finds no association between treatment and sex. Note that we got a p-value based on 10000 Monte carlo replications.

Specification of strata requires careful thought. For instance, if there are too many strata and each patient ends up in his or her own stratum uniquely specified by a set of prognostic factors, the result is equivalent to having used simple randomization. Plans for randomization must take this possible problem into account and use only a few strata, relative to the size of the trial. In other words, most blocks should be filled because unfilled blocks permit imbalances.

Also, if the block sizes are too large, the probability of assigning treatment A or B will be approximately equal to 50% for all the early assignments in a block (since N_A and N_B will be large). This is essentially equivalent to using simple randomization, except for the assignments towards the end of the block.

5 Urn schemes

As the number of draws increases, the addition of a single ball has a smaller effect and the proportion of red and white balls in the urn will converge to 50%. Thus, the procedure will behave like a simple 1:1 randomization scheme.

First, we are going to use the library `randomizeR`. The function `udPar` specifies the desired design. We tell `R` to start with 1 ball for each treatment and randomly draw a ball from the urn. Then, 1 ball is added to the urn from the opposite treatment. This algorithm is easy to implement in `R` (see function `urnModel` below).

```
set.seed(12)
library(randomizeR)

## Loading required package:  ggplot2
## Loading required package:  plotrix

x = udPar(N=2500, ini = 1, add = 1, groups = LETTERS[1:2])
y = genSeq(x,r = 1,seed = 12)
y

##
## Object of class "rUdSeq"
##
## design = UD(1,1)
## seed = 12
## N = 2500
## groups = A B
## ini = 1
## add = 1
##
## The sequence M:
##
## 1 A   A   A   B   B   B   A   A   B   B   ...
```

Now we can get the list of assignments

```
c(getRandList(y))[1:10]

##  [1] "A" "A" "A" "B" "B" "B" "A" "A" "B" "B"

table(c(getRandList(y)))

##
##    A    B
## 1271 1229
```

```
urnModel = function(n)
 {
   na = 1
   nb = 1

   prob = rep(NA,n)
   trt = rep(NA,n)
   u = runif(n)

   for (i in 1:n)
   {
     # Probability of getting A
     prob[i] = nb/(na+nb)

     if (u[i]<prob[i])
     {
       trt[i] = "A"
       na = na + 1
     } else{trt[i] = "B";nb = nb + 1}
   }

   return(data.frame(treatment = trt,probA = prob,na = na,nb = nb))
 }

draws = urnModel(n = 2500)
head(draws)

##   treatment     probA   na   nb
## 1         B 0.5000000 1241 1261
## 2         A 0.6666667 1241 1261
## 3         A 0.5000000 1241 1261
## 4         A 0.4000000 1241 1261
## 5         B 0.3333333 1241 1261
## 6         B 0.4285714 1241 1261

table(draws$treatment)

##
##    A    B
## 1240 1260
```

The plot is shown in the following figure: We clearly observe that the probability of picking a ball converges to 50%, as the number of draws increases.

6 Comparison of different randomization methods

We assume that a small clinical trial of $N = 14$ patients is to take place. In this case, the selection of the method of randomization is important since we might end up with different number of

```r
plot(1:2500,draws$probA,ylab = "Probability",xlab = "draws",
      main = "Probablity of getting A based on the urn model",type = "l")
abline(h = 0.5,col = "red")
```

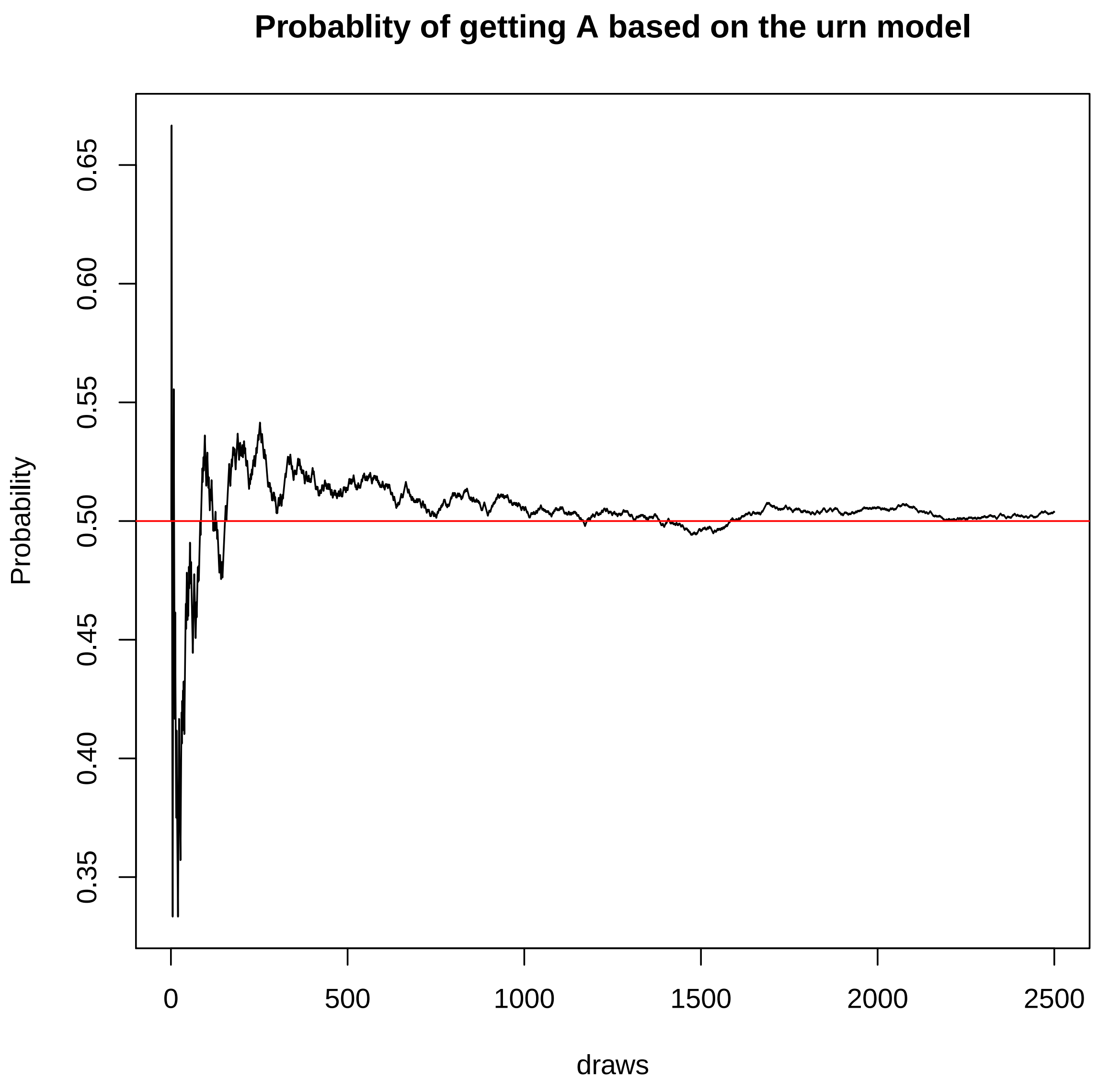

Figure 3: Long-term probability of treatment assignment in an urn scheme

subjects in each group. We are going to examine the properties of (1) simple randomization (2) randomization in blocks (3) urn models through simulation. The `R` code follows

```r
# Number of Subjects
set.seed(5)
N = 14
```

```r
it = 200

# Results
results = data.frame(SR = rep(NA,it),BR = NA,urn = NA)

for (i in 1:it)
 {
   ########################
   # Simple randomization #
   ########################
   data = data.frame(id = paste("id",1:N,sep = ""), trt = NA)

   # Simulate u to decide
   u = runif(N)

   # Assign patients to treatment arms
   data$trt[u<0.5] = "A"
   data$trt[u>0.5] = "B"

   results$SR[i] = table(data$trt)[1]/N

   ###########################
   # Randomization in blocks #
   ###########################
   block_rand = blockrand(n = N, num.levels = 2, levels = c("A","B"),
                          block.sizes = 1:4)[1:N,]

   results$BR[i] = table(block_rand$treatment)[1]/N

   ##############
   # Urn scheme #
   ##############
   data = urnModel(N)

   results$urn[i] = table(data$treatment)[1]/N
 }
```

Now let's see how they faired.

```r
# 50% chance of getting each treament in the long run
apply(results,2,mean)

##        SR        BR       urn
## 0.4989286 0.4989286 0.4967857
```

We see that all three methods have the same long-term behavior. However, not all three have the same variability (i.e., tendency to have long runs in either direction of 50%). We can see it with the following output:

```
# The variance is particularly high in simple randomization
apply(results,2,sd)

##         SR         BR        urn
## 0.13056944 0.03240402 0.08720218
```

We can also see it pictorially in the following figure: For all methods the probability of assigning

```
plot(cumsum(results$SR)/1:it,type = "l",lty = 1,col = "black",
      ylim = c(0.40,0.60),ylab = "Probability of getting A",xlab = "Draws")
lines(cumsum(results$BR)/1:it,lty = 2,col = "red")
lines(cumsum(results$urn)/1:it,lty = 3,col = "blue")

legend("topright",lty = 1:3,col = c("black","red","blue"),
       legend = c("Simple randomization","Randomization in blocks",
                  "Urn scheme"),bty = "n")
```

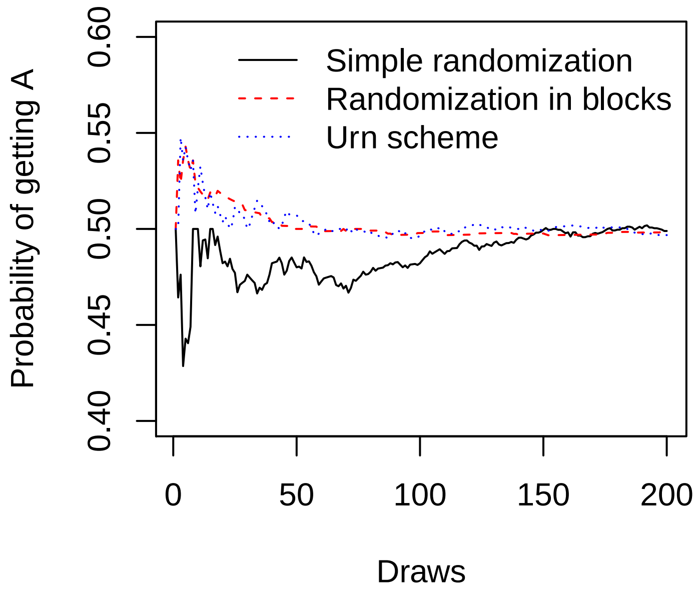

Figure 4: Long-term behavior of three randomization techniques

patients to treatment A is close to 50% in the long run. However, the variance of simple randomization is particularly high. For simple randomization, the average proportion of treatment A is not 50% even if we could repeat the randomization process about 50-100 times!