

InnoRand – Innovative Randomization

PHUSE US 2024 – SM08

Yinfei Wu | Boehringer Ingelheim Pharmaceuticals, Inc.
Ivo Marek | Entimo AG

Disclaimer

This presentation is provided 'as is' without any sort of warranty. The views expressed in this presentation are those of the speakers and not necessarily those of Boehringer Ingelheim or Entimo. Boehringer Ingelheim and Entimo do not provide any sort of guarantee with regards to the accuracy of the information provided in this presentation.

Agenda

1. The Lack of Randomization Methods
2. Permuted Block Design
3. Block Urn Design

Randomization Procedures

- Ideal: maximize the statistical power (balancing) and minimize selection bias
- No single randomization procedure meets the properties 'balancing' and 'selection bias control'
- The R package: randomizeR implements different methods
- Randomized permuted block design – block sizes and treatments are not drawn in alternation
- Block Urn Design is not implemented
- Goal: Extension of randomizeR on GitHub

Permuted Block Design

- M blocks containing n patients
- random allocation rule within a block to ensure balance ($n/2$ is the maximum imbalance)
- Example: two treatments 1 and 2, block size is 4
- Possible block allocations: 1122, 2211, 1212, 2121, 1221, 2112
- Usually `sample(c(1, 1, 2, 2))` is used
- natural course: draw the pattern itself
- `sample(list(c(1, 1, 2, 2), c(2, 2, 1, 1), ...), size = 1)[[1]]`

Permutations

- iterpc package iterates efficiently over permutations/combinations

- Create an iterator object:

```
I <- iterpc::iterpc(n = c(2,2), ordered = TRUE)
```

- Get all permutations: `perms <- iterpc::getall(I)`

- Sample one permutation: `perms[sample(1:6, size = 1),]`

- Problem: error for large block sizes

- five treatments $a = 1, 2, 3, 4, 5$ with multiplicity $m_a = 3, 4, 2, 3, 5$

-> 1715313600 permutations (> 100Gb memory)

Partitioning

- Solution: Build $A-1$ permutation tables
- $$\binom{n}{m_1 \ m_2 \ \dots \ m_A} = \frac{n!}{m_1 \ m_2 \ \dots \ m_A} = \binom{n}{m_1} \binom{n-m_1}{m_2} \binom{n-m_1-m_2}{m_3} \dots \binom{m_A}{m_A}$$
- Define: $s[a] = s_a = m_{a+1} + m_{a+2} + \dots + m_A$ for $a < A$ and $s[A] = s_A = 0$
- Multinomial Coefficient rewritten:
$$\binom{s_1+m_1}{m_1} \binom{s_2+m_2}{m_2} \binom{s_3+m_3}{m_3} \dots \binom{s_A+m_A}{m_A}$$
- Define: $\text{bin}[a] = \binom{s_a+m_a}{m_a}$
- Weights: $w[a] = \text{bin}[a+1] * \text{bin}[a+2] * \dots * \text{bin}[A]$ for $a = 1, 2, \dots, A-1$

Step 1 - Permutation Tables

- multiplicities (m_a, s_a) with labels $(a, 0)$ for $a < A - 2$ and $(a, a + 1)$ for $a = A - 1$
- Initialization: `A <- length(m); bin <- c(); i <- 0L`
- ```
s <- rev(cumsum(rev(m[-1L])))
perms <- lapply(m[-A], function(x) {
 a <- a+1L
 if (a < A-1) l <- c(a, 0L) else l <- c(a, a+1L)
 I <- iterpc::iterpc(c(x, s[a]), ordered = TRUE, labels = l)
 bin <- c(bin, iterpc::getlength(I))
 iterpc::getall(I)
})
bin <- c(bin, 1L)
w <- rev(cumprod(rev(bin[-1L])))
```

## Step 2 - Randomization

- **Sample:** `R = sample(0L: (prod(bin)-1L), size = 1L)`
- `a <- 0L`  
`X <- sapply(w, function(x) {`  
    `a <- a+1L`  
    `R[a+1L] <- R[a] %% w[a]`  
    `1L + (R[a] - R[a+1L]) / w[a]`  
`})`
- Example: `bin = c(680, 1001, 45, 56, 1)` and `w = c(2522520, 2520, 56, 1)`
- $R[1] = 1071253633$  is randomly drawn from  $\{0, 1, 2, \dots, 1715313599\}$
- $R[2] = R[1] \bmod w[1] = 1071253633 \bmod 2522520 = 1705153$
- $X[1] = 1 + (R[1]-R[2]) / w[a] = 1 + 1069548480 / 2522520 = 425$

## Step 3 - Combine

- Select from Table a the  $X[a]$ -th record and combine in reverse order:
- ```
block <- perms[[A-1L]][X[A-1L],]  
for (a in (A-2L):1L) {  
  block <- replace(perms[[a]][X[a],], perms[[a]][X[a],]==0L, block)  
}
```
- Example:
- $X[4] = 10$ (10-th permutation): 4 5 4 5 5 5 4 5
- $X[3] = 30$ (30-th permutation): 0 0 0 3 0 0 0 0 0 3 -> 4 5 4 3 5 5 5 4 5 3
- $X[2] = 677$: 0 0 0 2 2 2 0 0 0 0 0 2 0 0 -> 4 5 4 2 2 2 3 5 5 5 4 2 5 3
- $X[1] = 425$: 0 0 0 0 1 0 0 0 1 1 0 0 0 0 0 0 0 -> 4 5 4 2 1 2 2 3 1 1 5 5 5 4 2 5 3

Intro Block Urn Design

- **BUD's Origin:** This method was first proposed by Zhao & Weng in 2011
- **Current State:** No existing R packages support
- **BUD's Advantages:**
 - **Consistent Imbalance Control:** BUD ensures a balanced distribution across different groups.
 - **Reduced Deterministic Assignments:** BUD lowers the chances of predictable assignments, enhancing the randomness of the trial.
 - **Applicability:** BUD is versatile and can be applied to trials with more than two treatments.

Zhao W, Weng Y. Block urn design - a new randomization algorithm for sequential trials with two or more treatments and balanced or unbalanced allocation. Contemp Clin Trials. 2011 Nov;32(6):953-61. doi: 10.1016/j.cct.2011.08.004. Epub 2011 Aug 22. PMID: 21893215; PMCID: PMC3206733.

The Mechanism of Block Urn Design

Start with an "urn" (a list or array) containing "balls" (allocation chances) for each group



When a participant is enrolled, draw a ball to determine their group treatment assignment



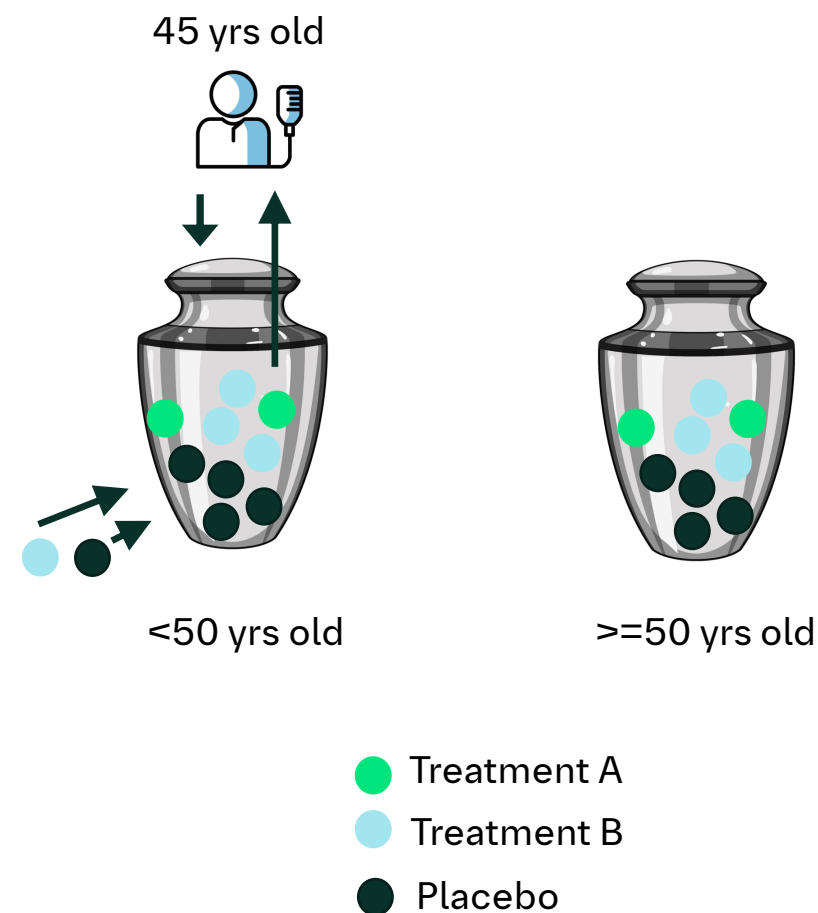
Adjust the contents of the urn based on the assignment to maintain balance over time



The "block" aspect of BUD allows for controlling allocation within certain blocks or strata of participants

BUD in Action: A Hypothetical Example

- Clinical trial with three treatment groups: Drug A, Drug B, and a placebo
- Aim to ensure each group has a balanced number of participants
- Use BUD to adapt allocation based on participant characteristics (e.g., age, gender, disease severity)
- Separate urns may be used for different age groups or disease severity levels to ensure balance within these strata



Outlook

- Library will be embedded in a GxP compliant R Shiny App
- Extension of the Library

Thank You!

Yinfei Wu

Boehringer Ingelheim Pharmaceuticals, Inc.

Email: yinfei.wu@boehringer-ingelheim.com

Web: www.boehringer-ingelheim.com

Ivo Marek

Entimo AG

Email: ima@entimo.de

Web: www.entimo.com