Random Assignment Subject to Constraints

Alexander Coppock, Yale University

6/18/2017

Most random assignment schemes can be handled by the `randomizr` package for R as a variant of simple, complete, clustered, or blocked random assignment. However, some assignment schemes can't be expressed as one (or even a combination) of these. In this document, I show how one such scheme can be expressed as a linear programming problem. I am grateful to Robin, a student in my 2016 Field Experiments course for stumping me on a random assignment protocol, her friend Kat who figured it out in Excel (shudder) using solver, and to this Stack Overflow answer (https://stackoverflow.com/a/35805005/4172083) for inspiration.

Standard Random Assignment Procedures

Most random assignment schemes can be expressed as either simple or complete:

1. Simple: "Coin Flip" random assignment. Subjects are independently assigned to conditions.
2. Complete: "Shuffled Deck" random assignment. A fixed set of treatment conditions is randomly allocated to a fixed number of subjects.

Cluster and Block random assignment can be seen as special cases of simple or complete random assignment:

- Cluster random assignment is a special case of either simple or complete random assignment in which whole groups of subjects ("clusters") are assigned to conditions together.
- Blocked random assignment is a special case of complete random assignment in which blocks of subjects are completely randomly assigned to conditions.

These procedures are all easily handled by the `randomizr` package for R, using the `simple_ra`, `complete_ra`, `cluster_ra`, `block_ra` functions.

```r
library(randomizr)
Z_simple <- simple_ra(N = 100)
Z_complete <- complete_ra(N = 100)

clust_var <- rep(letters, times = 1:26)
Z_cluster <- cluster_ra(clust_var = clust_var)

block_var <- rep(c("A", "B","C"), times = c(50, 100, 200))
Z_block <- block_ra(block_var = block_var)
```

The Problem

Imagine a dataset of 80 fingers on 16 hands belonging to 8 people. Our goal is assign 40 fingers to treatment and 40 to control. We want to treat exactly half the thumbs, indexes, etc. We also want to treat exactly 5 fingers for each person. And we want each hand to have two or three treated fingers.

At first this seems like a blocking problem. With blocked random assignment, we conduct complete random assignment within each block. But if we create blocks of person-hand-finger, we have 80 unique blocks of size 1. If we create person-finger blocks, we have 40 unique blocks, but we have no guarantee that each hand will have either two or three treated fingers.

What we need to do is to impose a series of constraints and let the `lpSolve` package for R solve the linear programming problem.

The Solution

First, let's make a dataset that corresponds to an 8-person version of the experiment:

```
library(lpSolve)
library(tidyverse)

N_people <- 8

subjects <-
  expand.grid(person = paste0("person_", 1:N_people),
              finger = c("thumb", "index", "middle", "ring", "pinky"),
              hand = c("right", "left")) %>%
  mutate(person_finger = paste0(person, "_", finger),
         person_hand = paste0(person, "_", hand),
         finger_hand = paste0(finger, "_", hand))

head(subjects)
```

```
##     person finger  hand  person_finger    person_hand finger_hand
## 1 person_1  thumb right person_1_thumb person_1_right thumb_right
## 2 person_2  thumb right person_2_thumb person_2_right thumb_right
## 3 person_3  thumb right person_3_thumb person_3_right thumb_right
## 4 person_4  thumb right person_4_thumb person_4_right thumb_right
## 5 person_5  thumb right person_5_thumb person_5_right thumb_right
## 6 person_6  thumb right person_6_thumb person_6_right thumb_right
```

We are going need access to the unique "person-finger", "person-hand", and "finger-hand" combinations

```
unique_person_fingers <- unique(subjects$person_finger)
unique_person_hands <- unique(subjects$person_hand)
unique_finger_hands <- unique(subjects$finger_hand)

N_person_fingers <- length(unique_person_fingers)
N_finger_hands <- length(unique_finger_hands)
N_person_hands <- length(unique_person_hands)
```

We're going to solve an optimization problem subject to three constraints.

1. We treat exactly 1 of each "person_finger". (One of person 1's pinkies, one of person 2's indexes, etc.)
2. We treat 2 or more fingers on each "person_hand" type (Each hand has to have at least two treated fingers)
3. Exactly half of each "finger_hand" is treated. (Half of the left indexes is treated, etc.)

The `lp` function needs the `const.mat`, `const.dir`, and `const.rhs` arguments specified in a particular fashion, which is what we are making here:

```
first <- t(sapply(unique_person_fingers, function(i) as.numeric(subjects$person_finger == i)))
second <- t(sapply(unique_person_hands, function(i) as.numeric(subjects$person_hand == i)))
third <- t(sapply(unique_finger_hands, function(i) as.numeric(subjects$finger_hand == i)))

const.mat <- rbind(first, second, third)
```

```r
const.dir <- rep(c("=", ">=", "="), c(N_person_fingers, N_person_hands, N_finger_hands))
const.rhs <- rep(c(1, 2, N_people/2), c(N_person_fingers, N_person_hands, N_finger_hands))
```

Now that we have set the constraints, we need to introduce some randomness. The `random_objective` object is where the actual stochastic component of this procedure enters.

```r
# This step makes it stochastic...
random_objective <- runif(ncol(const.mat))

mod <- lp(
  direction = "max",
  objective.in = random_objective,
  const.mat = const.mat,
  const.dir = const.dir,
  const.rhs = const.rhs,
  all.bin = TRUE
)

subjects$Z <- mod$solution
table(subjects$Z)
```

```
##
##  0  1
## 40 40
```

We can check that indeed, we have satisfied the goals.

```r
with(subjects, table(person, Z))
```

```
##           Z
## person     0 1
##   person_1 5 5
##   person_2 5 5
##   person_3 5 5
##   person_4 5 5
##   person_5 5 5
##   person_6 5 5
##   person_7 5 5
##   person_8 5 5
```

```r
with(subjects, table(hand, Z))
```

```
##        Z
## hand     0  1
##   right 20 20
##   left  20 20
```

```r
with(subjects, table(finger, Z))
```

```
##         Z
## finger   0 1
##   thumb  8 8
##   index  8 8
##   middle 8 8
##   ring   8 8
##   pinky  8 8
```

```r
with(subjects, table(person_hand, Z))
```

```
##                Z
## person_hand      0 1
##   person_1_left  3 2
##   person_1_right 2 3
##   person_2_left  2 3
##   person_2_right 3 2
##   person_3_left  2 3
##   person_3_right 3 2
##   person_4_left  3 2
##   person_4_right 2 3
##   person_5_left  2 3
##   person_5_right 3 2
##   person_6_left  2 3
##   person_6_right 3 2
##   person_7_left  3 2
##   person_7_right 2 3
##   person_8_left  3 2
##   person_8_right 2 3
```

```r
with(subjects, table(finger_hand, Z))
```

```
##              Z
## finger_hand    0 1
##   index_left   4 4
##   index_right  4 4
##   middle_left  4 4
##   middle_right 4 4
##   pinky_left   4 4
##   pinky_right  4 4
##   ring_left    4 4
##   ring_right   4 4
##   thumb_left   4 4
##   thumb_right  4 4
```

```r
with(subjects, table(person_finger, Z))
```

```
##                  Z
## person_finger     0 1
##   person_1_index  1 1
##   person_1_middle 1 1
##   person_1_pinky  1 1
##   person_1_ring   1 1
##   person_1_thumb  1 1
##   person_2_index  1 1
##   person_2_middle 1 1
##   person_2_pinky  1 1
##   person_2_ring   1 1
##   person_2_thumb  1 1
##   person_3_index  1 1
##   person_3_middle 1 1
##   person_3_pinky  1 1
##   person_3_ring   1 1
##   person_3_thumb  1 1
```

```
##    person_4_index  1 1
##    person_4_middle 1 1
##    person_4_pinky  1 1
##    person_4_ring   1 1
##    person_4_thumb  1 1
##    person_5_index  1 1
##    person_5_middle 1 1
##    person_5_pinky  1 1
##    person_5_ring   1 1
##    person_5_thumb  1 1
##    person_6_index  1 1
##    person_6_middle 1 1
##    person_6_pinky  1 1
##    person_6_ring   1 1
##    person_6_thumb  1 1
##    person_7_index  1 1
##    person_7_middle 1 1
##    person_7_pinky  1 1
##    person_7_ring   1 1
##    person_7_thumb  1 1
##    person_8_index  1 1
##    person_8_middle 1 1
##    person_8_pinky  1 1
##    person_8_ring   1 1
##    person_8_thumb  1 1
```

Conclusion

The reason we need this procedure is that in a setup like this one, each "subject" is a unique snowflake - Person 1 only has one left pinky, one right index, etc. If she had *two* left pinkies, then we could block on `person_hand_finger`, and randomly assign exactly one to treatment every time. In the absence of multiple left pinkies per person, we want to assign either the left or the right pinky to treatment. But we *also* want to ensure that exactly half of the left pinkies are assigned to treatment. These complex and overlapping constraints mean that we have to abandon the standard blocking paradigm. Instead, we can express the problem as a linear programming optimization scheme where the stochasticity comes from the random "utility" in the objective function.