Specimen Randomization in Longitudinal Biomarker Studies: Constrained Pocock Simon Randomization

Michael C. Donohue

August 29, 2013

1 Introduction

In longitudinal or crossover biomarker studies in which the biological assays must be run in batches, i.e. arrays or plates, it is often desirable to allocate all of the longitudinal samples from a given individual to the same batch. Biological assays can be expensive, so there is often a desire to utilize as much of each batch as possible. Different numbers of longitudinal samples per individual are also common, further complicating the allocation constraints. We propose a constrained Pocock Simon (1975) randomization routine for these scenarios.

Recall the standard Pocock Simon randomization routine works by assessing the potential imbalance for each of M prognostic factors if the next subject were randomized to each of N groups. We call this *imbalance* measure d_{ik}, $i = 1, \ldots, M$, $k = 1, \ldots, N$. The *total imbalance* $G_k = G(d_{1k}, \ldots, d_{Mk})$ aggregates the imbalance across the M factors for each of the group allocations.

The randomization probabilities, p_k, can either be a fixed value or a function of G_k. For our setting, we propose a simple modification to allow the p_k to also be a function of the number of longitudinal samples available, and the maximum samples per batch, so that it is not possible over allocate to a given batch.

2 Example

Suppose we have the following subjects to randomize:

```
library(SRS)
data(srs_data)
head(srs_data)

  ID Counts Group   Age
1  1      5     1   old
2  2      5     1   old
3  3      5     1   old
4  4      5     1 young
5  5      5     1   old
6  6      4     1 young
```

where `Counts` represents the number of longitudinal samples that we have for each subject. We wish to randomly allocate each subject to one of 13 plates, maintaining a good balance of `Group` (5 levels) and `Age` (2 levels).

```
p.func.greedy.if.possible <- function(overallImbalance, treatmentCounts, maxCounts)
{
    cant.go <- treatmentCounts > maxCounts
    if(all(cant.go))
```

```
      stop("Randomization impossible. Probably need another treatment group.")

    number.of.treatments <- length(overallImbalance)
    k <- which(overallImbalance == min(overallImbalance))
    p.vec <- rep(0, number.of.treatments)
    p.vec[k] <- 1
    p.vec/sum(p.vec)
    p.vec[cant.go] <- 0

    if(all(p.vec == 0)){ # try less greedy
      number.of.treatments <- length(overallImbalance)
      p.star <- 2/3
      k <- which(overallImbalance == min(overallImbalance))
      if (length(k) > 1) {
          k <- sample(k, 1)
      }
      p.vec <- rep((1 - p.star)/(number.of.treatments - 1), number.of.treatments)
      p.vec[k] <- p.star
      p.vec
      p.vec[cant.go] <- 0
    }

    p.vec
}

get.counts <- function(object)
{
  expt <- object@expt
  treatment.names <- expt@treatment.names
  factor.names <- expt@factor.names
  factor.level.names <- expt@factor.level.names
  treatment.names <- expt@treatment.names
  state.matrix <- object@stateTable
  tr.ratios <- object@tr.ratios

  tr.assignments <- object@tr.assignments
  tr.assignments$Treatment <- factor(tr.assignments$Treatment,
    levels = treatment.names)
  tr.assignments$Counts <- factor(tr.assignments$Counts,
    levels = factor.level.names[[which(factor.names == "Counts")]])
  tr.assignments <- with(tr.assignments, table(Counts, Treatment)) *
    as.numeric(factor.level.names[[which(factor.names == "Counts")]])
  colSums(tr.assignments)
}

expt <- ClinicalExperiment(number.of.factors = 3,
  factor.names = c('Counts', 'Group', 'Age'),
  number.of.factor.levels = c(2, 5, 2),
  factor.level.names =
    list(c(4, 5), 1:5, c('young', 'old')),
  number.of.treatments = 13,
  treatment.names = as.character(1:13))
```

```
g.func <- function(imbalances)
{
    factor.weights <- c (1, 100, 1)
    imbalances %*% factor.weights
}

r.obj <- new("cPocockSimonRandomizer", expt, as.integer(20130827),
  g.func=g.func, p.func = p.func.greedy.if.possible, max.counts = 30)

for(i in 1:nrow(srs_data)){
  r.obj <- randomize(r.obj, as.character(srs_data[i, "ID"]),
     as.character(srs_data[i, expt@factor.names]))
}

tr.assignments <- r.obj@tr.assignments
tr.assignments$Treatment <- factor(tr.assignments$Treatment,
  levels = r.obj@expt@treatment.names)
```

Sample counts (subjects × timepoints/infusions) per plate:

```
get.counts(r.obj)

 1  2  3  4  5  6  7  8  9 10 11 12 13
27 27 27 29 29 30 30 27 30 29 27 27 30
```

Distribution of sample counts per subject per plate

```
with(tr.assignments, table(Counts, Treatment))

      Treatment
Counts 1 2 3 4 5 6 7 8 9 10 11 12 13
     4 3 3 3 6 6 5 5 3 5  6  3  3  5
     5 3 3 3 1 1 2 2 3 2  1  3  3  2
```

Distribution of Group per plate

```
with(tr.assignments, table(Group, Treatment))

     Treatment
Group 1 2 3 4 5 6 7 8 9 10 11 12 13
    1 1 1 1 1 1 1 1 2 1  1  1  1  1
    2 1 1 1 1 1 1 1 1 2  1  1  1  1
    3 1 1 0 0 0 0 1 0 0  0  1  0  1
    4 1 1 1 1 1 1 0 1 1  1  1  1  0
    5 2 2 3 4 4 4 4 2 3  4  2  3  4
```

Distribution of Age per plate

```
with(tr.assignments, table(Age, Treatment))

       Treatment
Age     1 2 3 4 5 6 7 8 9 10 11 12 13
  old   3 3 3 3 3 2 3 3 4  4  3  2  3
  young 3 3 3 4 4 5 4 3 3  3  3  4  4
```