Tim B.

Marginal sampling within `sceptre`

Definition and motivation

I describe "inductive sampling without replacement," a procedure for sharing without replacement (WOR) samples across gene-gRNA pairs containing the same number of negative control (NT) cells but different numbers of treatment cells. Inductive sampling WOR reduces the number of WOR samples that must drawn by a factor equal to the number of gRNAs in the dataset. On a more technical level inductive sampling WOR considerably reduces the number of database queries (specifically, gene expression vector loads) that must be issued when running SCEPTRE out-of-core. Inductive sampling WOR therefore could reduce compute by several orders of magnitude.

Let N be the number of control cells. Label the control cells by $c_1, c_2, \ldots, c_N$. Next, let M be the number of treatment cells containing the the gRNA that infects the greatest number of cells. Label the treatment cells by $t_1, \ldots, t_M$. We seek to construct a length-M random sequence $a_1, a_2, \ldots, a_M$ that satisfies the following properties:

1. $a_i \in \{c_1, \ldots, c_N, t_1, \ldots, t_i\}$ for all a_i (i.e., the ith element of the sequence is a control cell or one of the first i treatment cells).

2. $a_i \neq a_j$ (i.e., the elements of the sequence are unique).

3. Letting A_i denote the set containing the first i elements of the sequence (i.e., $A_i := \{a_1, a_2, \ldots, a_i\}$), then

$$\mathbb{P}(c_1 \in A_i) = \mathbb{P}(c_2 \in A_i) = \cdots = \mathbb{P}(c_N \in A_i) = \mathbb{P}(t_1 \in A_i) = \cdots = \mathbb{P}(t_i \in A_i) = \frac{i}{i+N}.$$

In other words, we seek to construct a sequence of increasing sets $A_1 \subset A_2 \subset \cdots \subset A_M$ that satisfies an "inductive" WOR sampling property. The first set A_1, which contains a single element, contains the control cells and the first treatment cell with equal probability. The second set A_2, which contains two elements, contains the control cells and each of the first *two* treatment cells with equal probability, and so on. This property is appealing because

it enables us to share random samples across gRNAs. If a given gRNA has i treatment cells, then A_i is a valid WOR sample for this gRNA. Thus, we need only generate a random sequence $a_1 \dots a_M$ once and share this random sequence across all gRNAs.

Constructing an inductive WOR sample

I describe a strategy for constructing an inductive WOR sample, and I prove its correctness. I describe the procedure inductively.

Step 1. Sample one element from the set $a_1 \leftarrow \{c_1, \dots, c_N, t_1\}$, putting an equal mass of $1/(N+1)$ onto each of the elements.

Step i, for $i \geq 2$. Let $B_i := \{c_1, \dots, c_N, t_1 \dots, t_{i-1}\} \setminus A_{i-1}$ be the set of "leftover" elements that were not selected in step $i-1$. There are N elements in the set B_i. Draw an element at random from the set $B_i \bigcup \{t_i\}$, placing a mass of $\frac{i}{i+N}$ on t_i and a mass of $\left(1 - \frac{i}{i+N}\right)/N$ on each of the elements in B_i. Set the sampled element to a_i. Continue this process until $i = M$, resulting in a sequence $a_1, \dots, a_M$. This sequence satisfies the desired property stated above, which I now prove.

Proof. I proceed inductively. Base case: Let $i = 1$. Then $\mathbb{P}(c_1 \in A_1) = \dots \mathbb{P}(c_N \in A_1) = \mathbb{P}(t_1 \in A_1) = 1/(1+N)$. Next, let $i \in \{2, \dots, M\}$ be given. Inductive step: Suppose that

$$\mathbb{P}(c_1 \in A_i) = \dots = \mathbb{P}(c_N \in A_i) = \mathbb{P}(t_1 \in A_i) = \dots = \mathbb{P}(t_i \in A_i) = \frac{i}{N+i}.$$

We construct a_{i+1} by sampling t_{i+1} with probability $(i+1)/(N+i+1)$ and the elements of B_i with probability $\left(1 - \frac{i+1}{N+i+1}\right)/N$. We have by construction that

$$\mathbb{P}(t_{i+1} \in A_{i+1}) = \frac{i+1}{N+i+1}.$$

Next, let $u \in \{c_1, \dots, c_N, t_1, \dots, t_i\}$. We compute $\mathbb{P}(u \in A_{i+1})$ using the law of total probability:

$$\mathbb{P}(u \in A_{i+1}) = \mathbb{P}(u \in A_{i+1} | u \in A_i)\mathbb{P}(u \in A_i) + \mathbb{P}(u \in A_{i+1} | u \notin A_i)\mathbb{P}(u \notin A_i). \tag{1}$$

Considering first the lefthand term, we have that $\mathbb{P}(u \in A_{i+1} | u \in A_i) = 1$, as membership in A_i implies membership in A_{i+1}. Next, $\mathbb{P}(u \in A_i) = i/(N+i)$

by the inductive hypothesis. Thus, the left term of (1) is $i/(N+i)$. Next, consider the righthand term. If $u \notin A_i$, then $u \in B_k$. Thus,

$$\mathbb{P}(u \in A_{i+1} | u \notin A_i) = \left(1 - \frac{i+1}{N+1+i}\right)/N.$$

Furthermore, by the inductive hypothesis, $\mathbb{P}(u \notin A_{i-1}) = 1 - i/(N+i)$. Stringing these pieces together,

$$\begin{aligned}
\mathbb{P}(u \in A_{i+1}) &= \frac{i}{N+i} + \frac{1}{N}\left(1 - \frac{i+1}{N+1+i}\right)\left(1 - \frac{i}{N+i}\right) \\
&= \frac{i}{N+i} + \frac{1}{N}\left(\frac{N+1+i}{N+1+i} - \frac{i+1}{N+1+i}\right)\left(\frac{N+i}{N+i} - \frac{i}{N+i}\right) \\
&= \frac{i}{N+i} + \frac{1}{N}\left(\frac{N}{N+1+i}\right)\left(\frac{N}{N+i}\right) = \frac{i}{N+i} + \left(\frac{1}{N+1+i}\right)\left(\frac{N}{N+i}\right) \\
&= \frac{i}{N+i} + \frac{N}{(N+1+i)(N+i)} = \frac{i(N+1+i)+N}{(N+i)(N+1+i)} = \frac{iN+i+i^2+N}{(N+1)(N+1+i)} \\
&= \frac{(N+i)(i+1)}{(N+1)(N+1+i)} = \frac{i+1}{N+1+i}.
\end{aligned}$$

□

An algorithm for inductive WOR sampling

I describe an algorithm for inductive WOR sampling using only a $U(0,1)$ random number generator. First, I introduce a distribution — the "IWOR distribution." The IWOR(N,i) distribution is a discrete probability distribution that has support $\{0,\ldots,N\}$ and places mass $i/(i+N)$ on N and mass $\frac{1}{N}\left[1 - i/(i+1)\right]$ on $\{0,1,\ldots,N-1\}$. One can sample from the IWOR(N,i) distribution as follows (Algorithm 1). The algorithm is fast, requiring only an if statement, a multiplication, and a floor operation.

Next, I give an algorithm for constructing an inductive WOR sequence $a_1 a_2 \ldots a_M$ given negative control cells $c_1, \ldots c_N$ and treatment cells $t_1, \ldots, t_M$ (Algothtm 2; the algorithm uses zero-based indexing.) The algorithm is efficient, requiring $O(M)$ time and $O(N+M)$ space. In practice we will index the control cells by $0, 1, \ldots, N-1$ and the treatment cells by $N, N+1, \ldots, N+M-1$. Thus the initialization of r step (namely, line 2) can be

rewritten as

$$r \leftarrow [0, 1, \ldots, N-1, N],$$

and the final line of the for loop can be rewritten as

$$r[N] \leftarrow N + i - 1.$$

Suppose we want to construct a sequence $a_1 a_2 \ldots a_{m-1} a_m a_{m+1} \ldots a_{M-1} a_M$ that satisfies the inductive WOR property from m to M. (In our application m and M may be the minimum and maximum number of treatment cells, respectively). We can construct $a_1 \ldots a_{m-1}$ via standard sampling WOR and then run Algorithm 2 to construct $a_m \ldots a_M$. A common algorithm for standard WOR sampling is the Fisher-Yates sampler.

Algorithm 1 Sampling from the IWOR(N, i) distribution.

Require: N, i
  $u \sim U(0, 1)$
  $p \leftarrow i/(N + i)$
  if $u > 1 - p$ **then**
    $d \leftarrow N$
  else
    $d \leftarrow \lfloor uN/(1-p) \rfloor$ // floor operator
  end if
  return d

Algorithm 2 Constructing an inductive WOR sample.

Require: Control cells $c_1, \ldots, c_N$ and treatment cells $t_1, \ldots, t_M$.
  Initialize $v \leftarrow \text{vector}(M)$.
  Initialize $r \leftarrow [c_1, c_2, \ldots, c_N, t_1]$.
  for i = 1 ... M **do**
    `pos` $\sim$ IWOR(N, i) // sample a position within v
    $v[i-1] \leftarrow r[$`pos`$]$ // extract the element at that position
    $r[$`pos`$] \leftarrow r[N]$ // move the rightmost entry of r to position `pos`
    $r[N] \leftarrow t_{i+1}$ // update the rightmost entry of r with t_{i+1}
  end for
  return v

Algorithm 3 Hybrid Fisher-Yates/IWOR sampler

Require: Control cells $c_1, \ldots, c_N$ and treatment cells $t_1, \ldots, t_M$; the minimum number of treatment cells m.

$x \leftarrow [c_1, \ldots, c_N, t_1, \ldots, t_m]$.
$v \leftarrow \text{vector}(M)$
for $i = 1, \ldots, m$ **do** // Fisher-Yates step:
    $u \sim \text{Unif}(\{0, 1, \ldots, N + m - i\})$
    $\text{Swap}(x[N + m - i], x[u])$
end for
// entries $x[0, \ldots, N - 1]$ are N leftovers from x
// entries $x[N, \ldots, N + m - 1]$ are m samples WOR from x

for $i = 0, \ldots, m - 1$ **do**
    $v[i] \leftarrow x[i + N]$
end for

$x[N] \leftarrow t_{m+1}$ // initialize inductive WOR step
for i $= m + 1, \ldots, M$ **do**
    `pos` $\sim$ IWOR(N, i)
    $v[i - 1] \leftarrow x[\texttt{pos}]$
    $x[\texttt{pos}] \leftarrow x[N]$
    $x[N] \leftarrow t_{i+1}$
end for
return v

Sampling without replacement for the undercover analysis

The undercover analysis differs from the discovery analysis in low MOI. In the undercover analysis the total number of cells remains the same, while the number of treatment and control cells varies. Index the negative control cells by $\{1, \ldots, N\}$. Denote the number of cells in the largest undercover group by M. We seek to construct a length-M sequence of indices with the following property:

1. $a_i \in \{1, \ldots, N\}$ for all a_i. (Note: there is no restriction on the specific values that a_i can take as a function of i; instead, a_i can be any member of $\{1, \ldots, N\}$).
2. $a_i \neq a_j$.
3. Let $A_i = \{a_1, \ldots, a_i\}$. Then

$$\mathbb{P}(1 \in A_i) = \mathbb{P}(1 = A_i) = \cdots = \mathbb{P}(N \in A_i) = i/N.$$

We can use a length-M sequence $a_1 a_2 \ldots a_M$ to generate WOR samples for gRNAs in the undercover analysis. For example, suppose that a given undercover gRNA is present in $k < M$ cells. To test this gRNA, we need to randomly assign a treatment to k of the cells in $\{1, \ldots, N\}$.

MOI	**Sampling mechanism**	**Analysis type**	**Sampling algorithm**
Low	Marginal	Disovery	Inductive WOR
Low	Marginal	Undercover	Standard WOR
Low	Conditional	Discovery	?
Low	Conditional	Undercover	?
High	Marginal	Discovery	Standard WOR
High	Marginal	Undercover	Standard WOR
High	Conditional	Discovery	?
High	Conditional	Undercover	?

Table 1: Resampling algorithms to use in different settings. The inductive WOR algorithm is used only in the following setting: low MOI, marginal resampling, discovery analysis. It is unclear what algorithm to use when resampling *conditional* (as opposed to *marginally*).