Urn Sampling Without Replacement: Enumerative Combinatorics In R

Robin K. S. Hankin
Auckland University of Technology

Abstract

This vignette is based on Hankin (2007).

This short paper introduces a code snippet in the form of two new R functions that enumerate possible draws from an urn without replacement; these functions call C code, written by the author. Some simple combinatorial problems are solved using the software.

For reasons of performance, this vignette uses pre-calculated answers. To calculate everything from scratch, set variable `calculate_from_scratch` in the first chunk to `TRUE`.

Keywords: Urn problems, drawing without replacement, enumerative combinatorics, Scrabble, R.

1. Introduction

Drawing balls from an urn without replacement is a classical paradigm in probability (Feller 1968). It is useful in practice, and many elementary statistics textbooks use it to introduce the binomial and hypergeometric distributions.

In this paper, I introduce software, written by the author, that enumerates[1] all possible draws from an urn containing specified numbers of balls of each of a finite number of types. Order is not important in the sense that drawing AAB is equivalent to drawing ABA or BAA. Formally, drawing n balls without replacement from an urn containing $f_1, f_2, \ldots, f_S$ balls of types $1, 2, \ldots, S$ is equivalent to choosing a solution $a_1, \ldots, a_S$ to the Diophantine equation

$$\sum_{i=1}^{S} a_i = n, \qquad 0 \leqslant a_i \leqslant f_i, \tag{1}$$

with probability $\left(\begin{array}{c} f_i \\ a_i \end{array} \right) \Big/ \left(\begin{array}{c} \sum f_i \\ n \end{array} \right)$.

Combinatorial enumeration is often necessary in the context of integer optimization: the appropriate configurations are enumerated and the optimal one reported. Sometimes explicit enumeration is needed to count solutions satisfying some condition: simply enumerate candidate solutions, then test them one by one.

The software discussed here comprises two new R (R Development Core Team 2008) functions `S()` and `blockparts()`, which are currently part of the **partitions** package (Hankin 2005),

[1] Enumerate: "to mention [a number of things] one by one, as if for the purpose of counting"

version 1.3-3. These functions call C code, also written by the author, which is available as part of the package.

All software is available from CRAN, https://CRAN.r-project.org/.

2. Examples

The software associated with this snippet is now used to answer a variety of combinatorial questions, written in textbook example style, that require enumerative techniques to solve.

Question A chess player is considering endgames in which White has a king, no pawns, and exactly three other pieces. What combinations of white pieces are possible? No promotions have occurred.

Answer This is an urn problem with a pool of 7 objects, in this case non-king chess pieces. An enumeration of the size-3 draws is required, which is given by new function `blockparts()`. This function enumerates the distinct solutions to equation 1 in columns which appear in lexicographical order:

```
> blockparts(c(Bishops=2,Knights=2,Rooks=2,Queens=1),3)

Bishops 2 1 2 1 0 1 0 2 1 0 1 0 0
Knights 1 2 0 1 2 0 1 0 1 2 0 1 0
Rooks   0 0 1 1 1 2 2 0 0 0 1 1 2
Queens  0 0 0 0 0 0 0 1 1 1 1 1 1
```

The first sample appears as the first column: this is the first lexicographically, as all draws are from as low an index of `f` as possible. Subsequent draws are in lexicographical order. Starting with a draw `d`—a vector of `length(y)` elements—the next draw is obtained by the following algorithm:

1. Starting at the beginning of the vector, find the first block that can be moved one square to the right, and move it. If no such block exists, all draws have been enumerated: **stop**.
2. Take all blocks to the left of the one that has moved, and place them sequentially in the frame, starting from the left.
3. Go to item 1.

Figure 1 shows an example of this in action. The left diagram shows a draw of a bishop and two knights, corresponding to the the second column of the matrix returned by `blockparts()` above. The first block that can be moved is one of the knights. This moves one place to the right and becomes a rook. The remaining pieces (that is, one bishop and one knight) are redistributed starting from the left; they become two bishops.

There are thus 13 combinations: note that the majority of them have no Queen. The solutions are in lexicographical order, which is useful in some contexts.

Question "Scrabble" is a popular word board game that involves choosing, at random, a rack of 7 tiles from a pool of 100 with the following frequencies:

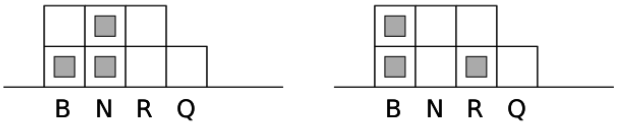

Figure 1: A pictorial description of the algorithm used in function `blockparts()`. Three blocks (grey squares) are arranged in a tableau (B, N, R, Q, representing the chess pieces under consideration) in two consecutive configurations, the left one first. The larger, line squares above each piece name show the maximum number of chess pieces allowed; thus the two knights in the left diagram completely fill the 'N' column and this indicates that a maximum of two knights may be drawn. The left diagram thus corresponds to one bishop and two knights: this is column two in the matrix returned by `blockparts()` in the R chunk above. The right diagram shows the next lexicographical arrangement, corresponding to column three; the algorithm for the change is described in the text

```
> scrabble
```

```
 a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p  q  r  s  t  u  v  w  x  y  z
 9  2  2  4 12  2  3  2  9  1  1  4  2  6  8  2  1  6  4  6  4  2  2  1  2  1  2
```

(note the last entry: two of the tiles are blank).

(***i***) how many distinct racks are possible?

(***ii***) what proportion of racks have no blanks?

(***iii***) what is the most probable rack and what is its frequency?

Answer

(***i***). The number of draws is given by function `S()`. This function returns the number of solutions to equation 1 by determining the coefficient of x^n in the generating function $\prod_{i=1}^{S}\sum_{j=0}^{f_i} x^j$ using the **polynom** (Venables, Hornik, and Maechler 2006) package:

```
> S(scrabble,7)
```

```
[1] 3199724
```

(***ii***). The number of racks with no blank is given by function `S()`, applied with a suitably shortened vector argument; the proportion of racks with no blank is then:

```
> S(scrabble[-27],7)/S(scrabble,7)
```

```
[1] 0.7763745
```

Note that this question is distinct from that of determining the *probability* of drawing no blanks, which is given by elementary combinatorial arguments as $\binom{98}{7}\Big/\binom{100}{7}$, or about 86%. This value is larger because racks with one or more blanks have relatively low probabilities of being drawn.

(***iii***). To determine the most probable rack, we note that the probability of a given draw is given by

$$\frac{\prod\binom{f_i}{a_i}}{\binom{100}{7}}.$$

The appropriate R idiom would be to enumerate all possible racks using `blockparts()`, and apply a function that calculates the probability of each rack:

```
> f <- function(a) { prod(choose(scrabble, a))/choose(sum(scrabble), 7) }
> racks <- blockparts(scrabble, 7)
> probs <- apply(racks, 2, f)
```

The draw of maximal probability is given by the maximal element of `probs`. The corresponding rack is then:

```
> rep(names(scrabble), racks[, which.max(probs)])

[1] "a" "e" "i" "n" "o" "r" "t"
```

In the context of the full Scrabble problem, there is only one acceptable anagram of the most probable rack: "otarine". Its probability is

```
> max(probs)

[1] 0.0001049264
```

or just over once per 9531 draws. It is interesting to note that the *least* probable rack is not unique: there are exactly 1469 racks each with minimal probability (about 6.247×10^{-11}).

3. Conclusions

The software discussed in this code snippet enumerates the possible draws from an urn made without replacement; it is used to answer several combinatorial questions that require enumeration for their answer. Further work might include enumeration of solutions of arbitrary linear Diophantine equations.

Acknowledgement

I would like to acknowledge the many stimulating and helpful comments made by the R-help list while preparing this software.

I would also like to thank an anonymous referee who suggested that the `polynom` package could be used to evaluate the generating function appearing in `S()`.

References

Feller W (1968). *An Introduction to Probability Theory and its Applications*, volume 1. Third edition. New York: Wiley.

Hankin RKS (2005). "Additive integer partitions in R." *Journal of Statistical Software, Code Snippets*, **16**(1).

Hankin RKS (2007). "Urn sampling without replacement: enumerative combinatorics in R." *Journal of Statistical Software, Code Snippets*, **17**(1).

R Development Core Team (2008). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL https://www.R-project.org.

Venables B, Hornik K, Maechler M (2006). ***polynom**: A Collection of Functions to Implement a Class for Univariate Polynomial Manipulations*. R package version 1.2-1. S original by Bill Venables, packages for R by Kurt Hornik and Martin Maechler., URL https://CRAN.R-project.org/package=polynom.

Affiliation:

Robin K. S. Hankin
Auckland University of Technology
AUT Tower
Wakefield Street
Auckland, New Zealand
E-mail: hankin.robin@gmail.com