Set Partitions in R

Robin K. S. Hankin Auckland University of Technology

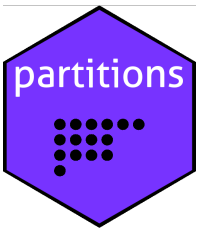
Luke J. West National Oceanography Centre, Southampton

Abstract

This vignette is based on Hankin (2007a).

This short paper introduces a code snippet in the form of an R function that enumerates all possible partitions of a finite set given the sizes of the equivalence classes. Three combinatorial problems are solved using the software: one from bioinformatics, one from scheduling theory, and one from forensic science.

Keywords: Set Partitions, Forensic Science, Enumerative Combinatorics, R.



1. Introduction

A *partition* of a set $S = \{1, 2, \ldots, n\}$ is a family of sets $T_1, T_2, \ldots T_k$ satisfying

1. $T_i \cup T_j = \emptyset$ if $i \neq j$
2. $\bigcup_{i=1}^{k} T_k = S$
3. $T_i \neq \emptyset$ for $i = 1, \ldots, k$.

There are exactly fifteen ways to partition a set of four elements, shown in Table 1 using standard notation. Here (13)(2)(4) represents the same partition as (2)(4)(31): the induced equivalence relation is identical. Either one may be represented as `1213`, indicating that elements one and three are in the first equivalence class ("1"), element two in the second, and element four in the third. In Table 1, two partitions appear in the same column if their 'shape' (that is, the size distribution of the equivalence classes, formally defined in Section 2 below) is the same. The shapes correspond to integer partitions in standard form (Hankin 2005), and appear in standard order (Andrews 1998). This suggests a natural method for enumerating the equivalence relations on a set of cardinality n: enumerate the integer partitions of n, and for each of these, enumerate the distinct equivalence relations of that shape.

This paper introduces software that enumerates[1] all set partitions of a set of a specified (finite)

[1] The transitive verb "to enumerate" can be used in two senses: firstly, "to calculate the number of"; and secondly, "to list each occurrence of, as if for the purpose of counting". This paper adopts the second usage.

(1234)	(123)(4)	(12)(34)	(12)(3)(4)	(1)(2)(3)(4)
	(124)(3)	(13)(24)	(13)(2)(4)	
	(134)(2)	(14)(23)	(14)(2)(3)	
	(234)(1)		(23)(1)(4)	
			(24)(1)(3)	
			(34)(1)(2)	

Table 1: The fifteen partitions of a set of four elements

size. The new functionality presented here is given by `setparts()`, an R (R Development Core Team 2007) wrapper for C++ code, currently part of the **partitions** package (Hankin 2005, 2007b), version 1.7-3, available under the GPL from CRAN, `http://www.r-project.org/`.

Function `setparts()` takes a single argument `a`, coerced to integer mode. The default case is that of `a` being a vector of length greater than one, in which case it returns all partitions of a set of `sum(a)` elements into equivalence classes of the sizes of the elements of `a`; in the case of `a` being of length one, it returns all partitions of a set of `a` elements. If `a` is a matrix, it returns all equivalence classes with sizes of the columns of `a`.

2. Algorithms

In this paper, we present an algorithm that lists all partitions of a given "shape". Formally, the **shape** $\lambda = (\lambda_1, \ldots \lambda_k)$ of a partition is given by the sizes of the T_i; these are conventionally given in standard (i.e. non-increasing) order. Thus λ is an integer partition of n (Hankin 2005), written $\lambda \vdash n$.

The distinct equivalence relations may be enumerated, for a given integer partition, using a recursive algorithm that operates on partially filled partitions.

Consider an integer partition $\lambda = (\lambda_1, \ldots \lambda_k)$, also written $\left(1^{f_1}, \ldots, n^{f_n}\right)$; the latter notation indicates that f_i elements of λ are equal to i. The algorithm used here fills the equivalence classes sequentially from left to right using a technique originally due to Killworth (2007).

The partitions are ordered from largest to smallest; non-empty ties are broken lexicographically. This prevents the reporting of identical equivalence classes (to see this, consider the two identical equivalence relations (123)(45)(67)(8) and (123)(67)(45)(8): the second descriptor does not occur in the algorithm because the third equivalence class precedes the second).

Thus, starting with an empty partition of a given size, one may apply the algorithm detailed in Figure 1 recursively until each full partition is recorded.

2.1. Performance and complexity

The total number of set partitions corresponding to a particular integer partition λ is given by elementary combinatorial arguments as

$$\frac{n!}{\prod_{i=1}^{k} \lambda_i! \cdot \prod_{j=1}^{n} f_j!}. \tag{1}$$

However, the number of integer partitions increases rapidly with n; the exact value is given by the partition function `P(n)` of the package (Hankin 2005), but the asymptotic form given

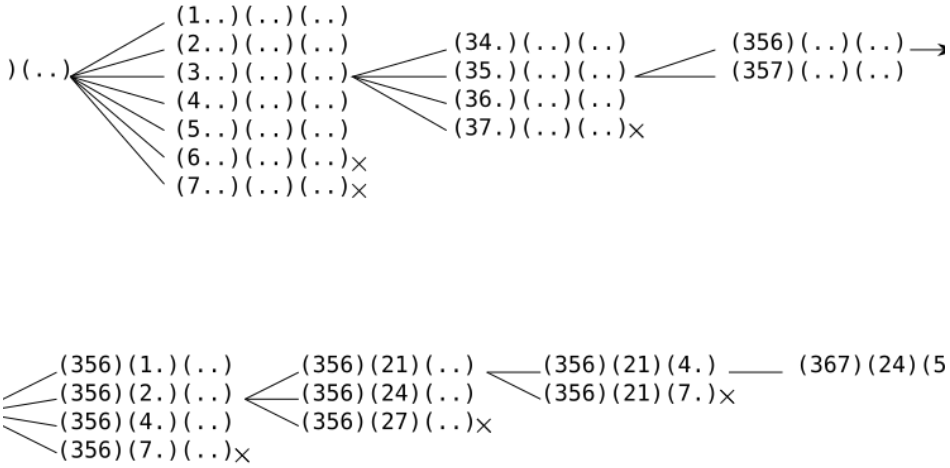


Figure 1: The algorithm used to enumerate set partitions of a given shape. Unfilled positions are indicated by a dot. Of the partitions at a given level, the children of one are shown. The initial empty configuration is filled from left to right from a 'pool' of unused elements; equivalence classes are ordered by size (largest first) and ties are broken lexicographically at runtime to suppress reporting of identical equivalence relations in which two classes are transposed. Within a class, the elements of partially- and completely- filled equivalence classes are ordered in increasing order. A cross indicates an illegal configuration in which the partially filled class cannot be completed as insufficient elements remain in the pool; the algorithm prunes such configurations as soon as they are detected. Successful termination of the algorithm, viz. a filled partition, is indicated by a tick ("check mark") at which point the filled equivalence class is reported, and the recursion bottoms out

by Hardy and Ramanujan (1918), viz.

$$P(n) = \frac{e^{\pi\sqrt{2n/3}}}{4n\sqrt{3}} \cdot \left(1 + O(n^{-1})\right) \tag{2}$$

shows that the rate of growth is rapid. The number of *set* partitions of a set of size n is given by the Bell numbers $B(n)$ (Rota 1964), which grow much more rapidly than the partition function; an asymptotic form was given by Bender (1974):

$$B(n) = t^{n-t} e^{t-1} (\log t)^{-1/2} \cdot (1 + o(n)) \tag{3}$$

where $t = e^{W_0(n-1/2)}$. Here $W_0(\cdot)$ denotes the principal branch of the Lambert W function (Corless, Gonnet, Hare, Jeffrey, and Knuth 1996), provided with the **gsl** package (Hankin 2006). Equation 3 is easily evaluated and can be used to give a feel for the magnitude of the Bell numbers. Note that convergence is quite slow; for example, $B(10) = 115975$ (compare about 1.48×10^3 from Equation 3), and $B(100) \simeq 4.76 \times 10^{115}$ (compare 5.43×10^{115}).

3. Applications

Many problems in science and industry involve the optimization of some desideratum over a finite number of options. Such optimization is frequently soluble within the discipline of computational combinatorics: simply enumerate the possible solutions and choose the best. One advantage of this method over heuristic optimization techniques such as tabu search (Glover 1997) is that by enumerating all possible solutions, one is *certain* to find the global optimum. In this section we present three examples of `setparts()` in use: one from bioinformatics, one from multiprocessor scheduling, and one from forensic science.

3.1. Bioinformatics

In studies of allelic segregation under disomic inheritance, one situation encountered is that of enumerating the possible arrangements of alleles on genetic loci[2]. Rodzen and May (2002) consider the white sturgeon; there is much uncertainty surrounding the organization of this species's genome.

Many alleles of interest may be identified in the genome of the white sturgeon. Sometimes these are known to be either monosomic or disomic, but it is not known which alleles are colocated, nor which alleles are monosomic and which are disomic. If researchers have identified, say, five alleles (conventionally labelled 1-5), and further that two pairs of these are colocated at two disomic loci and one is monosomic, then there are a number of possible arrangements for these alleles on the chromosome. In the language of the package, we wish to enumerate the partitions of a set of five elements (the alleles) into equivalence classes of size 2,2,1 (the loci).

[2] A *chromosome* is a double helix of DNA, typically occurring in matched pairs in cell nuclei. A *locus* is an identifiable position on a chromosome pair; a *gene* is a distinct sequence of DNA base pairs occurring at a locus; an *allele* is one of a number of distinguishable forms for a particular gene; the *genome* is the complete set of genes present in the chromosomes; a *monosomic locus* contains two copies of one, and a *disomic locus* two different, alleles. Two alleles at the same locus are said to be *colocated*. A monosomic (disomic) allele is one that occurs at a monosomic (disomic) locus in a given genome. A *gamete* is a sequence of alleles chosen from the genome: monosomic and disomic alleles are chosen with probability 1 and 0.5 respectively.

The five alleles are arranged into two from one disomic locus, two from another disomic locus, and one from a monosomic locus.

Enumerating the possible arrangements is accomplished using the `setparts()` function:

```
> setparts(c(2,2,1))

[1,] 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3
[2,] 2 2 3 1 1 1 2 2 3 2 2 3 1 1 1
[3,] 3 2 2 3 2 2 1 1 1 3 2 2 2 1 2
[4,] 2 3 2 2 3 2 3 2 2 1 1 1 2 2 1
[5,] 1 1 1 2 2 3 2 3 2 2 3 2 1 2 2
```

Each column comprises two 1s, two 2s, and one 3. Thus the first column, `x=12321`, say, indicates that alleles 1 and 5 [`which(x==1)`] occur at the first locus (which is disomic); alleles 2 and 4 [`which(x==2)`] at the second (also disomic); and allele 3 [`which(x==3)`] at the third locus, which is monosomic. This could be written (15)(24)(3); the R idiom would be

```
> a <- c(1,2,3,2,1)
> split(seq_along(a),a)

$`1`
[1] 1 5

$`2`
[1] 2 4

$`3`
[1] 3
```

[use `listParts(c(2,2,1))` to see all possible equivalence classes]. Two examples of electrophoresis gels, corresponding to the partitions of the first and last columns respectively, are shown in figure 2, which is analogous to Figure 1 of Rodzen and May (2002).

3.2. Scheduling theory

The "augmented multiprocessing model" is an abstract description of a certain class of problems that occurs frequently in the field of scheduling theory (Garey and Johnson 1975). We consider a simple variant of the augmented multiprocessing model (specifically, the "three partition problem", which is known to be NP-complete) and solve it using the software presented here.

Consider a set of tasks $\mathcal{T} = \{T_1, \ldots, T_m\}$; task i requires time τ_i for its completion in the sense that if it begins executing at time t it will complete at time $t + \tau_i$. Each of these tasks is to be executed by one of n processors. It is desired to allocate the tasks among the processors so that all tasks are completed in the minimum possible time.

Thus if there are $m = 4$ tasks with T_i requiring $\tau_i = i$ time units ("seconds") for completion, and $n = 2$ processors, it is clear that allocating tasks 1 and 4 to one processor, and tasks 2

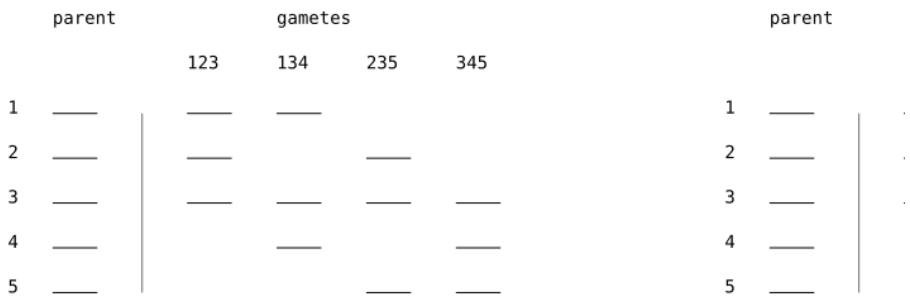


Figure 2: Two electrophoresis gels showing the possible bands that can occur with two different arrangements of alleles. In both gels, two loci are disomic, and one is monosomic. Left, gel corresponding to (15)(24)(3), or `12321`; this would mean that alleles 1 and 5 are at one disomic locus, alleles 2 and 4 at another, and allele 3 at a monosomic locus. Thus band combinations 1-5 and 2-4 are absent. The gel on the right corresponds to (1)(24)(35), or `31212`

and 3 to the second is optimal. This would correspond to the set partition (14)(23); the time required is $1+4=2+3=5$ seconds. Other allocations such as (24)(13) would take longer ($2+4=6$ seconds).

More complex examples are not so straightforward. Consider the case where $m=9$ and again $\tau_i = i$; now there are $n=3$ processors. The possible task schedules may be enumerated using the `restrictedparts()` function[3] of the **partitions** package:

```
> jj <- setparts(restrictedparts(m,n,include.zero=FALSE))
> summary(jj)
```

```
 [1,] 1 1 1 1 1 1 1 1 1 ... 1 1 1 1 1 1 1 1 1
 [2,] 1 1 1 1 1 1 1 1 1 ... 2 2 2 2 2 2 2 2 2
 [3,] 1 1 1 1 1 1 1 1 1 ... 3 2 2 2 2 3 3 3 3
 [4,] 1 1 1 1 1 1 1 1 1 ... 3 3 3 3 2 2 2 2 3
 [5,] 2 2 2 2 1 1 1 1 1 ... 3 3 3 2 3 3 3 2 2
 [6,] 3 1 1 1 2 2 2 1 1 ... 2 3 2 3 3 3 2 3 3
 [7,] 1 3 1 1 3 1 1 2 2 ... 2 2 3 3 3 2 3 3 2
 [8,] 1 1 3 1 1 3 1 3 1 ... 1 1 1 1 1 1 1 1 1
 [9,] 1 1 1 3 1 1 3 1 3 ... 1 1 1 1 1 1 1 1 1
```

Each column corresponds to a task allocation schedule. The first column, for example, indicates that tasks 1,2,3,4,7,8,9 are allocated to processor 1, task 5 to processor 2, and task 6 to processor 3 and the set partition might be written (1234789)(5)(6). In the matrix, the maximum entry of each column is 3, showing that all three processors are always used (setting `include.zero` to `TRUE` would relax this requirement and allow some columns to correspond to allocations in which only one or two processors are allocated a nonzero number of tasks).

The objective function to be minimized is just the time for the processor which finished last:

```
> tau <- 1:9
> slowest <-  function(x) max(tapply(tau, x, sum))
```

The minimal objective function is clearly 45/3=15. Identifying allocations that attain this bound is straightforward:

```
> time.taken <- apply(jj,2,slowest)
> minimal.time <- sum(tau)/n
> jj[,time.taken == minimal.time]
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
[1,]    1    1    1    1    1    1    2    1    1
[2,]    1    1    1    1    2    2    1    2    2
[3,]    1    1    2    2    1    1    1    3    3
[4,]    1    2    1    2    1    2    1    3    2
[5,]    1    2    2    1    2    1    2    1    3
```

[3] Recall that `restrictedparts(a,b)` enumerates the possible partitions of `a` into at most `b` parts.

```
[6,]    2    2    3    3    3    1    1    2    1
[7,]    3    3    2    1    1    3    3    2    3
[8,]    3    3    1    2    2    3    3    3    1
[9,]    2    1    3    3    3    2    2    1    2
```

Thus 9 allocations attain the maximal possible efficiency: a conclusion difficult to achieve without enumeration. These optima may be differentiated using some secondary characteristic, such as even distribution of the *number* of tasks among the processors. This would prescribe either the eighth or ninth columns, which both allocate three tasks to each processor.

3.3. Forensic science

In the field of forensic science, one situation sometimes encountered is the following. Crimes C_j, where $i = 1, \ldots, n$ have been committed; it is desired to infer how many perpetrators are responsible and which crimes each perpetrator is responsible for. One prominent example would be multiple unsolved homicides (Roberts 2003).

Evidence at each of the n crime scenes is available in the form of m Bernoulli random variables that indicate various forensic characteristics at the crime scene (examples might include presence or absence of a weapon, forced entry, etc). The evidence may be organized in a m-by-n matrix E of zeroes and ones indicating whether forensic characteristic i was true at crime scene j, where $1 \leqslant i \leqslant m$ and $1 \leqslant j \leqslant n$.

A given row of E thus corresponds to a particular type of evidence; if perpetrator k committed crime j, we would have $E_{ij} = 1$ with probability p_{ik}, specific to perpetrator k. For each evidence i, it is assumed that each perpetrator has a p_{ik} chosen at random from a beta distribution with parameters α_i, β_i for $1 \leqslant i \leqslant m$, and that all perpetrators have the same α_i, β_i; these will be known (or at least estimated) from previous studies of similar crimes. An example is shown below.

```
          crime
evidence C1 C2 C3
      E1  0  0  1
      E2  0  0  1
      E3  0  0  1
      E4  1  1  0
      E5  1  1  0
```

Thus in this example there are $n = 3$ crimes and $m = 5$ forensic characteristics. Observe that `C1` and `C2` have identical evidence, and that `C3` has contrary evidence. This would suggest that there are exactly two perpetrators: one responsible for `C1` and `C2`, and one for `C3`, corresponding to partition (12)(3); the hypothesis that this is the case is written $H_{[1,1,2]}$.

Considering a single type of evidence, and a single perpetrator, the probability of observing a successes and b failures is

$$\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \int_{p=0}^{1} \left[\frac{(a+b)!}{a!b!} p^a (1-p)^b \right] p^{\alpha-1}(1-p)^{\beta-1}\, dp = \frac{(a+b)!}{a!b!} \cdot \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \cdot \frac{\Gamma(a+\alpha)\Gamma(b+\beta)}{\Gamma(a+b+\alpha+\beta)}.$$

Here the probability p is integrated over its range, weighted by its prior (the posterior would also be a beta distribution, with parameters $\alpha + a$ and $\beta + b$).

Now consider a set partition $\wp = T_1, \ldots, T_r$ of crimes; identify a perpetrator with each equivalence class of crimes and observe that a single perpetrator has m specific values of p_i, $1 \leqslant i \leqslant m$ and acts independently of other perpetrators. The likelihood of $\wp$ is then

$$\mathcal{L}(\wp) = C \left(\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \right)^r \cdot \prod_{k=1}^{r} \prod_{i=1}^{m} \frac{\Gamma(a_{ik}+\alpha_i)\Gamma(b_{ik}+\beta_i)}{\Gamma(a_{ik}+b_{ik}+\alpha_i+\beta_i)}$$

where $a_{ik} = \sum_{j \in T_k} E_{ij}$ and $b_{ik} = \sum_{j \in T_k} 1 - E_{ij}$ are the total successes and failures for evidence i due to perpetrator k (under $H_\wp$). Here C denotes an arbitrary multiplicative constant.

It is interesting to note that the case $\alpha = \beta = 1$ is not uninformative in this context, even though this induces a uniform prior distribution on p. To see this, consider the first line of the five by three matrix above `[0,0,1]`.

The likelihood function for this row is on five partitions:

$$\begin{aligned}
\mathcal{L}(H_{[1,1,1]}) &= \frac{\Gamma(\alpha+1)\Gamma(\beta+2)}{\Gamma(\alpha+\beta+3)} \cdot \left(\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \right)^1 \\
\mathcal{L}(H_{[1,2,1]}) &= \frac{\Gamma(\alpha+1)\Gamma(\beta+1)}{\Gamma(\alpha+\beta+2)} \cdot \frac{\Gamma(\alpha)\Gamma(\beta+1)}{\Gamma(\alpha+\beta+1)} \cdot \left(\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \right)^2 \\
\mathcal{L}(H_{[1,1,2]}) &= \frac{\Gamma(\alpha)\Gamma(\beta+2)}{\Gamma(\alpha+\beta+2)} \cdot \frac{\Gamma(\alpha+1)\Gamma(\beta)}{\Gamma(\alpha+\beta+1)} \cdot \left(\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \right)^2 \\
\mathcal{L}(H_{[2,1,1]}) &= \frac{\Gamma(\alpha+1)\Gamma(\beta+1)}{\Gamma(\alpha+\beta+2)} \cdot \frac{\Gamma(\alpha)\Gamma(\beta+1)}{\Gamma(\alpha+\beta+1)} \cdot \left(\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \right)^2 \\
\mathcal{L}(H_{[1,2,3]}) &= \frac{\Gamma(\alpha)\Gamma(\beta+1)}{\Gamma(\alpha+\beta+1)} \cdot \frac{\Gamma(\alpha)\Gamma(\beta+1)}{\Gamma(\alpha+\beta+1)} \cdot \frac{\Gamma(\alpha+1)\Gamma(\beta)}{\Gamma(\alpha+\beta+1)} \cdot \left(\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \right)^3 .
\end{aligned}$$

Thus if $\alpha = \beta = 1$ the likelihood function is

$$\begin{aligned}
\mathcal{L}(H_{[1,1,1]}) &= C/12 \\
\mathcal{L}(H_{[1,2,1]}) &= C/12 \\
\mathcal{L}(H_{[1,1,2]}) &= C/6 \\
\mathcal{L}(H_{[2,1,1]}) &= C/12 \\
\mathcal{L}(H_{[1,2,3]}) &= C/8
\end{aligned}$$

so the highest likelihood is indeed that of $H_{[1,1,2]}$. It is also instructive to assume $\alpha = \beta$ and consider the limit as $\alpha \longrightarrow 0$. This would correspond to perpetrators having either a very small p, or p close to 1, intermediate values being rare. To first order in α:

$$\begin{aligned}
\mathcal{L}(H_{[1,1,1]}) &= C \cdot \alpha/4 \\
\mathcal{L}(H_{[1,2,1]}) &= C \cdot \alpha/4 \\
\mathcal{L}(H_{[1,1,2]}) &= C \cdot (1-\alpha)/4 \\
\mathcal{L}(H_{[2,1,1]}) &= C \cdot \alpha/4 \\
\mathcal{L}(H_{[1,2,3]}) &= C \cdot 1/8
\end{aligned}$$

so the maximum likelihood partition would again be $H_{[1,1,2]}$; but one would not be able to reject the hypothesis that there are three distinct perpetrators (that is, $H_{[1,2,3]}$) at the two units of support level.

As a final example, consider the case where $\alpha = \beta$ and $\alpha \longrightarrow \infty$. This corresponds to each perpetrator leaving positive evidence with probability $\frac{1}{2}$. The five likelihoods all tend to the same limit.

Generalization to arbitrary factors

These ideas have a natural generalization to arbitrary factors. The beta distribution becomes a Dirichlet distribution and the posterior probability becomes

$$\frac{(\sum a_l)!}{\prod a_l!} \frac{\Gamma\left(\sum \alpha_l\right)}{\prod \Gamma\left(\alpha_l\right)} \frac{\prod \Gamma\left(a_l + \alpha_l\right)}{\Gamma\left(\sum a_l + \sum \alpha_l\right)}$$

where outcome l is observed with frequency a_l and the corresponding prior distributions are Dirichlet with parameters $\alpha_i = (\alpha_{i1}, \ldots, \alpha_{iu})$. The likelihood is then

$$\mathcal{L}(\wp) = C \prod_{i=1}^{n} \left[\left(\frac{\Gamma\left(\sum_l \alpha_{il}\right)}{\prod_l \Gamma\left(\alpha_{il}\right)} \right)^r \prod_{k=1}^{r} \frac{\prod_l \Gamma\left(a_{ikl} + \alpha_{il}\right)}{\Gamma\left(\sum_l \left(a_{ikl} + \alpha_{il}\right)\right)} \right] \tag{4}$$

where a_{ikl} is the frequency of outcome l for evidence i and perpetrator k, under $H_\wp$.

Example Consider a case where seven crimes have been committed; there are five forensic characteristics. The evidence matrix is as follows:

```
         crime
evidence C1 C2 C3 C4 C5 C6 C7
      E1  2  1  2  2  2  2  1
      E2  2  2  1  1  1  1  2
      E3  4  1  4  4  4  4  1
      E4  4  1  2  2  4  2  1
      E5  3  2  1  4  1  1  2
```

Each column corresponds to a crime and each row corresponds to a forensic characteristic. Rows 1 and 2 are Bernoulli: each is either 1 or 2. Rows 3-5 are Dirichlet with four possible outcomes. In this case, it is assumed that the prior is uniform, viz $\alpha_1 = \alpha_2 = 1$ for rows 1 and 2 and $\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = 1$ for rows 3-5.

Figure 3 shows the likelihood and support for each of the distinct set partitions on a set of six elements: variable `support` is from Equation 4. The maximum likelihood estimator is given by

```
> sp <- setparts(7)

> sp[,which.max(support)]

[1] 3 2 1 1 1 1 2
```

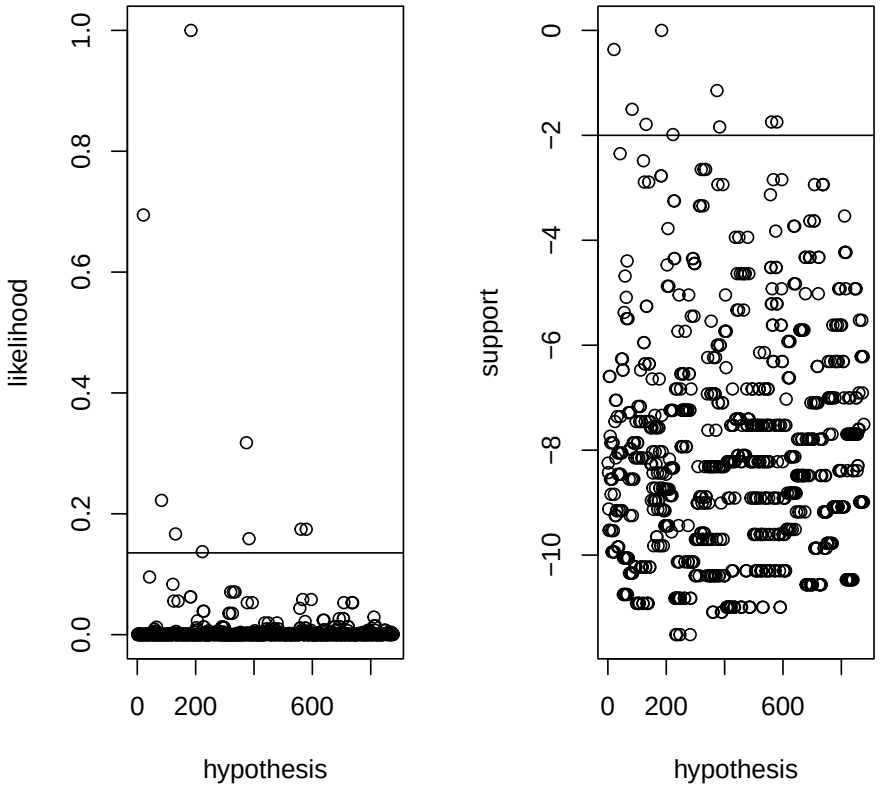


Figure 3: The likelihood (left) and support (right) for each of the 877 partitions on a set of seven elements. The horizontal line shows two units of support less than the maximum

which could be written (3456)(27)(1). This suggests that crimes 3-6 were committed by the same perpetrator, crimes 2 and 7 by another perpetrator, and crime 1 by a third perpetrator. Note that the maximum likelihood hypothesis is unique.

The likelihood function may be used to identify those hypotheses that cannot be rejected on the grounds of insufficient support. There are 9 partitions that have fewer than two units of support less than the maximum. These are:

```
> sp[, support > -2]
```

```
     partition
crime H1 H2 H3 H4 H5 H6 H7 H8 H9
   C1  3  1  2  2  3  3  1  2  2
   C2  2  2  3  2  2  2  2  3  3
   C3  1  1  1  1  1  1  1  1  1
   C4  1  1  1  1  1  4  3  2  1
   C5  1  1  2  1  4  1  1  1  1
   C6  1  1  1  1  1  1  1  1  1
   C7  2  2  3  2  2  2  2  3  4
```

and the supports for these hypotheses are

```
> support[support > -2]
```

```
[1]  0.0000000 -0.3646431 -1.1469859 -1.5040774 -1.7460665
[6] -1.7460665 -1.7917595 -1.8401331 -1.9853750
```

respectively.

Interpretation of this result is not straightforward. One may assert that crimes `C3` and `C6` were committed by the same perpetrator, and that certain pairs of crimes (specifically `C2-C3`, `C2-C4`, `C2-C5`, `C2-C6`, `C3-C7`, `C4-C7`, `C5-C7`, and `C6-C7`) were committed by different perpetrators: for any hypothesis that does not satisfy these criteria, one could gain at least two units of support by adopting instead the best supported hypothesis $H_{[3,2,1,1,1,1,2]}$, corresponding to (3456)(27)(1).

4. Note: a gotcha

Note carefully that `setparts(c(2,1,1))` does *not* enumerate the ways of placing four numbered balls in three labelled boxes of capacities 2,1,1. This is because there are two boxes of capacity 1, and swapping the balls between these boxes gives a different box partition but the same set partition (because sets are unordered).

```
> setparts(c(2,1,1))
```

```
[1,] 1 1 1 2 2 2
[2,] 2 1 2 1 1 3
[3,] 3 2 1 3 1 1
[4,] 1 3 3 1 3 1
```

Note the absence of a column reading `1 1 3 2`. This would correspond to placing balls 1 and 2 in box 1 (of size 2), ball 3 in box 3 (of size 1) and ball 4 in box 2 (also of size 1). The missing column is because the two boxes of size 1 are indistinguishable, so `1 1 3 2` is the same *set* partition as `1 1 2 3`, as in each case balls 3 and 4 are placed in "a box of size 1".

If you want to enumerate the ways of choosing two workers, a secretary and a chair from four people (who are conveniently numbered 1,2,3,4), use `multinomial()`:

```
> multinomial(c(worker=2,secretary=1,chair=1))

worker    1 1 1 1 1 1 2 2 3 2 2 3
worker    2 2 3 4 3 4 3 4 4 3 4 4
secretary 3 4 2 2 4 3 1 1 1 4 3 2
chair     4 3 4 3 2 2 4 3 2 1 1 1
```

Above, see how the rows are named for their equivalence class. The first two columns differ only in the identity of the secretary and the chair (and would be identical set partitions as given by `setparts(c(2,1,1))`). The reason I mention this is to avoid the following gotcha (which looks plausible but is incorrect):

```
> v <- c(worker=2,secretary=1,chair=1)
> a <- apply(setparts(v),2,order)
> rownames(a) <- rep(names(v),v)
> as.partition(a)

worker    1 1 1 2 2 3
worker    4 2 3 4 3 4
secretary 2 3 2 1 1 1
chair     3 4 4 3 4 2
```

The above is an incomplete enumeration. We see 1,4,2,3 [which would correspond to set partition 1,2,3,1] but not 1,4,3,2 [which would correspond to set partition 1,3,2,1]. These two differ in that the secretary and chair have swapped roles.

5. Conclusions

Enumeration of set partitions is required in several branches of computational combinatorics.

The software discussed in this code snippet enumerates set partitions for finite sets, under a variety of restrictions on the sizes of the induced equivalence classes and was used to solve three problems drawn from diverse applications in computational statistics.

Acknowledgement

We would like to acknowledge the many stimulating and insightful comments made on the R-help list while preparing this software.

References

Andrews GE (1998). *The Theory Of Partitions*. Cambridge University Press.

Bender EA (1974). "Asymptotic Methods in Enumeration." *SIAM Review*, **16**(4), 485.

Corless RM, Gonnet GH, Hare DEG, Jeffrey DJ, Knuth DE (1996). "On the Lambert W Function." *Advances in Computational Mathematics*, **5**, 329–359.

Garey MR, Johnson DS (1975). "Complexity Results for Multiprocessor Scheduling Under Resource Constraints." *SIAM Journal of Computing*, **4**(4), 397–411.

Glover F (1997). *Tabu Search*. Kluwer Academic.

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

Hankin RKS (2006). "Special Functions in R: Introducing the **gsl** Package." *R News*, **6**(4), 24–26. URL https://CRAN.R-project.org/doc/Rnews/.

Hankin RKS (2007a). "Set partitions in R." *Journal of Statistical Software, Code Snippets*, **23**(2).

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

Hardy GH, Ramanujan S (1918). "Asymptotic Formulæ in Combinatorial Analysis." *Proceedings of the London Mathematical Society*, **17**, 75–115. Series 2.

Killworth P (2007). Personal Communication.

R Development Core Team (2007). *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.

Roberts AR (2003). *Critical Issues in Crime and Justice*. Sage Publications.

Rodzen JA, May B (2002). "Inheritance of Microsatellite Loci in the White Sturgeon (*Acipenser Transmontanus*)." *Genome*, **45**, 1064–1076.

Rota GC (1964). "The Number of Partitions of a Set." *The American Mathematical Monthly*, **71**(5), 498–504.

Affiliation:

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

Luke J. West
National Oceanography Centre, Southampton
European Way

Southampton SO14 3ZH
United Kingdom