(Exercise 6)

- Particularly useful may be R's ability to edit just some entries of a matrix. Try the following representation of a 3-by-3 square inside a 6-by-6 one:

```
A <- matrix(0, nrow=6, ncol=6)
i <- 1
j <- 3
A[i:(i+2), j:(j+2)] <- 1
A
```

  To get you started with the second part of the exercise, suppose you have identified the 54 different smaller squares and written them out in the obvious order (smallest first, starting top left), then you could let

$$\{x_1, x_2, x_3, \ldots, x_{53}, x_{54}\}$$

  be your vector of optimization variables: any $x_i = 0$ means square i is not used, and any $x_i = 1$ means square i is used. Then it is possible to describe the constraint that exactly one square overlaps with the top left corner as follows:

$$x_1 + x_{26} + x_{42} + x_{51} = 1$$

  i.e. this matrix has a 1 in the top left corner

$$x_1 \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} + x_{26} \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} + x_{42} \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} + x_{51} \begin{pmatrix} 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

  You should find that if you manage to create all the individual squares first, then you can use them as columns of a first attempt at A.

- Later you may need some constraints to try and ensure that all your x_i variables are not larger than 1. This will further increase the size of A.
- Finally, if your code doesn't provide integer solutions (by construction you do need each $x_i \in \{0, 1\}$ not just $[0, 1]$) you may need to ask the 'lp' function to do some extra work for you. It is not actually easy to find only integer variable solutions – we will discuss it in a later exercise.

As usual, please do ask for help with syntax if you need it, but try not to ask for help with actual programming and coding ideas and methods, these are the things I want you to think about. This is not an easy exercise, it is designed to really get you thinking!

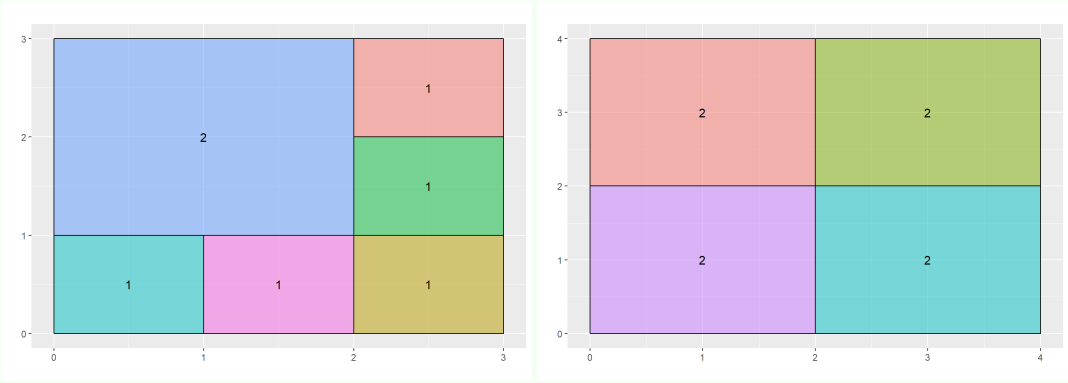
In this exercise you will need to create a relatively large number of constraints, and therefore also a relatively large A matrix and b vector. You can take advantage of the binding functions **cbind** and **rbind** which stick together vectors and matrices. For example, try the following:

```
A <- matrix(0, nrow=3, ncol=5)
B <- 1:5
rbind(A, B)
```

Exercise 6 (Ptolemy)**.** A famously difficult puzzle known as 'Squaring the square' involves trying to take an n-by-n square and decomposing it into smaller squares. Some versions of the problem demand the pieces are all different sizes, but we shall not consider that case. Our rules are as follows:

- For some integer n, start with an n-by-n square, composed of n^2 smaller squares.
- Your challenge is to find a partition of these n^2 squares into smaller squares using the fewest squares possible.

For example, for a 3-by-3 and 4-by-4 square the answers are fairly straightforward:



Your job is to write some code to find solutions like this. Try and find an optimal solution using a linear program for the 5-by-5 problem.

(a) Find the 54 squares of sizes 1, 2, 3 and 4, which lie inside a square of side 5.

(b) Create all 54 as separate matrices in R (put them in a list), store each of them as a 5-by-5 matrix full of zeroes except the positions occupied by the chosen square. For example, three of them will be

$$\begin{pmatrix}1&0&0&0&0\\0&0&0&0&0\\0&0&0&0&0\\0&0&0&0&0\\0&0&0&0&0\end{pmatrix}, \begin{pmatrix}0&0&1&1&0\\0&0&1&1&0\\0&0&0&0&0\\0&0&0&0&0\\0&0&0&0&0\end{pmatrix} \text{ and } \begin{pmatrix}0&0&0&0&0\\0&0&0&0&0\\1&1&1&0&0\\1&1&1&0&0\\1&1&1&0&0\end{pmatrix}$$

If you name all your matrices M_1, M_2, ..., M_{54} then your first 25 constraints can say $\sum_{i=1}^{54} x_i M_i$ equals a matrix full of 1's.

(c) Try to write a *linear program* such that your objective variables x_1, x_2, ..., x_{54} represent which of these matrices/squares are used. Each constraint should represent one of the 25 small squares, it may be useful to convert each M_i into a length 25 vector.

(d) Code up your linear program as a set of inputs, and use the solver (lp) to find an optimal solution. You will likely need to insert some additional constraints of the form $x_i \leq 1$ for each i.

(e) Did you need to use the 'int.vec' parameter in your work above? Did you otherwise get any strange non-integer solutions?

(Bonus question: Could you have written code for more general n-by-n rather than specifically 5-by-5?)