

# Latin Squares

## Objective:

To test if a given matrix is a latin square, to produce all possible latin squares of a given dimension and to investigate the connections between latin squares and group theory.

## Introduction:

An  $n \times n$  square matrix is a latin square of dimension  $n$  if the elements of the matrix are arranged in such a way that each is present exactly once in each row and each column. The elements of a latin square can be almost anything, including: numbers, letters, symbols and colored boxes. For this project, the scope will be limited to the integers from  $1 \dots n$ , where  $n$  is the dimension of the matrix. The following worksheet provides a means for checking whether or not a matrix is a latin square and discusses methods used to produce latin squares and their connection to group theory.

## Discussion:

### Part I: Testing a given matrix

The first step of the project consisted of writing a program that tests if an inputted matrix is a latin square. The program simply considers a matrix and returns a true if the matrix is a latin square or a false if it is not. The program tests if the integers ( $1 \dots n$ ) are all used and if each occurs exactly once in every row and column.

The program:

```
> islatin := proc(A)
  local i, n, v;
  if not type(A, 'matrix(integer, square)') then
    ERROR(`invalid arguments`)
  end if;

  n := linalg[rowdim](A);

  for v in [linalg[row](A, 1 .. n), linalg[col](A, 1
.. n)] do
    if convert(v, set) <> {seq(i, i = 1 .. n)} then
      RETURN(false)
    end if;
  end for;
end proc;
```

```

        end if;
    end do;
    RETURN(true)
end proc:

```

To use this procedure, a matrix must be inputted first with a name, and then checked using this program.

```

> a := matrix([[1, 2, 3, 4], [2, 3, 4, 1], [3, 4, 1, 2], [4,
1, 2, 3]]);

```

$$a := \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \\ 3 & 4 & 1 & 2 \\ 4 & 1 & 2 & 3 \end{bmatrix}$$

```

> islatin(a);

```

*true*

Therefore, this is an example of a matrix that is a latin square.

An example of a matrix that is not a latin square:

```

> b:= matrix([[4,4,3,3],[4,4,1,1],[3,3,1,1],[1,1,4,4]]);

```

$$b := \begin{bmatrix} 4 & 4 & 3 & 3 \\ 4 & 4 & 1 & 1 \\ 3 & 3 & 1 & 1 \\ 1 & 1 & 4 & 4 \end{bmatrix}$$

```

> islatin(b);

```

*false*

Matrix(b) above returns a false because every integer from 1 to n (here n=4) is not used and each integer is not limited to only one appearance in each row and column.

A non-square matrix returns an error because the dimensions do not fulfill the definition of a latin *square*.

```

> c:= matrix([[1, 2, 3],[3, 1, 2]]);

```

$$c := \begin{bmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \end{bmatrix}$$

```

> islatin(c);

```

Error, (in islatin) invalid arguments

>

[back to top](#)

## Part II: Producing all possible latin squares

The second part of the project consisted of writing a program that produces all possible latin squares of a given dimension, narrowing the vast number of possibilities by fixing the first row and column as 1...n in sequential order. As the dimension n gets larger, the number of possible latin squares grows rapidly. Without fixing the first row and column, there is one possible latin square for n=1, two possibilities for n=2, 12 possibilities for n=3, 576 possibilities for n=4, 161,280 possibilities for n=5, and so on (<http://mathworld.wolfram.com/LatinSquare.html>). The code that follows produces all of the latin squares with the first row and column set as 1...n. The above number of possibilities for each dimension of latin squares comes from the number of possibilities with the first row and column set multiplied by the number of permutations of the rows and columns. Thus, the number of latin squares produced by the following code multiplied by (n!)(n-1!) equals the total number of latin squares possible. Working backwards, this would indicate how many latin squares the following code should produce. If n=3 has a total of 12 latin squares, dividing that by (3!)(2!) would mean that the following program with n=3 would give just one latin square. For n=4,  $576/((4!)(3!)) = 4$ , so the program should produce four latin squares. For n=5,  $161,280/((5!)(4!)) = 56$ , so there should be 56 latin squares for the program's result. As you will see, this is true. Thus, the following program gives all of the latin squares with the first row and column set.

To make the program less difficult to follow, the actual procedure is broken down into smaller procedures. Thus, there are three smaller programs written that are used in the final procedure that produces the latin squares. These smaller programs only make sense in the context of the entire code, so here they are:

```
> listunion := proc(li)
  local C, i, L, m, set1, set2, set3;
    L := [];
    m := nops(li);
    for i from 1 to m do
      set1 := convert(L, set);
      set2 := convert(li[i], set);
      set3 := set1 union set2;
```

```

        L := convert(set3, list);
    end do;
    return L;
end proc:
> listminus := proc(list1, list2)
    convert(convert(list1, set) minus convert(list2, set),
list);
end proc:
> lsquare := proc(n, i, j)
    local B, C, d, k, r, x, z;
        B := matrix(n);
        x := linalg[rowdim](B);
        B[i,j] := [seq(k, k=1..x)];
        B[i,j] := listminus (B[i,j], linalg[row](n, i));
        B[i,j] := listminus (B[i,j], linalg[col](n,j));
        if B[i,j] = [] then
            return []
        end if;
        z := nops(B[i,j]);
        C := [seq(matrix(n), t=1..z)];
        for r from 1 to z do
            C[r][i,j] := B[i,j][r];
        end do;
        return C;
    end proc:

```

Here is the complete program that produces the latin squares, using the above three smaller procedures:

```

> latin := proc(n)
# n=row/column dimension
local C, g, i, j, x, z;
    x := matrix(n,n,0);
    for i from 1 to n do x[1,i] := i end do;
    for i from 1 to n do x[i,1] := i end do;
    C := [x];
    for i from 2 to n do
        for j from 2 to n do
            g := [seq(0, k=1..nops(C))];
            for z from 1 to nops(C) do
                g[z] := lsquare(C[z], i, j);
            end do;
            C := listunion(g);
        end do;
    end do;
    return C;
end proc:

```

To produce the latin squares of a certain dimension n, use latin(n):

```
> latin(2);
```

$$\begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$$

```
> latin(3);
```

$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \end{bmatrix}$$

```
> latin(4);
```

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 4 & 3 \\ 3 & 4 & 2 & 1 \\ 4 & 3 & 1 & 2 \end{bmatrix}, \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 4 & 1 & 3 \\ 3 & 1 & 4 & 2 \\ 4 & 3 & 2 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \\ 3 & 4 & 1 & 2 \\ 4 & 1 & 2 & 3 \end{bmatrix}, \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 4 & 3 \\ 3 & 4 & 1 & 2 \\ 4 & 3 & 2 & 1 \end{bmatrix}$$

These are all latin squares because the integers 1...n have all been used, and each integer appears exactly once in each row and column. The number of resulting latin squares corresponding with their dimension matches the predicted results from the opening comments. The above latin squares are all of the possible latin squares of their given dimension with the first row and column set. There is a problem with finding the latin squares of any dimension greater than 4, however.

```
> latin(5);
```

```
Error, (in latin) assigning to a long list, please use arrays
```

Based on preliminary calculations, this result should consist of 56 distinct latin squares. However, this is too much data for MAPLE to include in a list, so latin squares of dimension greater than 4 need to have their own special provisions made to the latin(n) procedure. This is difficult to do, so only latin(5) is rewritten here to show that it is possible. Anything beyond n=5 is really too big for MAPLE to handle using the methods used here with lists. It may be possible or more efficient to use arrays, but here the scope is limited to the possibilities while using lists.

For n=5, the procedure needs to be broken into parts. The first step within this new code uses latin(n) but changes the indices so the program just produces the possible squares with the first three rows completed. After this is done, new added steps run a loop through each of these possible latin squares individually to fill in the remaining rows. The procedure has been renamed latin5 so as not to confuse the two procedures.

```

> latin5 := proc()
  local C, F, g, i, j, k, n, S, t, x, z;
  # n is the dimension
  n := 5;
  x := matrix(n,n,0);
  for i from 1 to n do x[1,i] := i end do;
  for i from 1 to n do x[i,1] := i end do;
  C := [x];
  for i from 2 to 3 do
    for j from 2 to n do
      g := [seq(0, k=1..nops(C))];
      for k from 1 to nops(C) do
        g[k] := lsquare(C[k], i, j);
      end do;
      C := listunion(g);
    end do;
  end do;
  S := [];
  for z from 1 to nops(C) do
    F := [C[z]]:
    for i from 4 to n do
      for j from 2 to n do
        g := [seq(0, k=1..nops(F))];
        for k from 1 to nops(F) do
          g[k] := lsquare(F[k], i, j);
        end do;
        F := listunion(g);
      end do;
    end do;
    S := [op(S), F];
  end do;
  C := listunion(S);
end proc:

```

To produce all possible latin squares of dimension 5, use the command latin5(). Because there are going to be so many latin squares, and because it is important to know how many are actually produced to check this against preliminary calculations, naming the results "F" will help for later reference.

```
> F := latin5();
```

$$F := \left[ \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 5 & 1 & 3 & 4 \\ 3 & 4 & 2 & 5 & 1 \\ 4 & 3 & 5 & 1 & 2 \\ 5 & 1 & 4 & 2 & 3 \end{bmatrix}, \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 5 & 4 & 1 & 3 \\ 3 & 1 & 5 & 2 & 4 \\ 4 & 3 & 1 & 5 & 2 \\ 5 & 4 & 2 & 3 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 5 & 1 & 4 \\ 3 & 5 & 4 & 2 & 1 \\ 4 & 1 & 2 & 5 & 3 \\ 5 & 4 & 1 & 3 & 2 \end{bmatrix}, \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 5 & 4 & 3 & 1 \\ 3 & 4 & 1 & 5 & 2 \\ 4 & 1 & 5 & 2 & 3 \\ 5 & 3 & 2 & 1 & 4 \end{bmatrix} \right],$$



1 2 3 4 5	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
2 4 1 5 3	2 3 4 5 1	2 3 4 5 1	2 4 1 5 3
3 5 4 2 1	3 5 1 2 4	3 5 2 1 4	3 5 2 1 4
4 1 5 3 2	4 1 5 3 2	4 1 5 2 3	4 3 5 2 1
5 3 2 1 4	5 4 2 1 3	5 4 1 3 2	5 1 4 3 2
1 2 3 4 5	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
2 3 1 5 4	2 5 4 3 1	2 5 4 3 1	2 1 5 3 4
3 5 4 1 2	3 4 5 1 2	3 1 2 5 4	3 5 4 2 1
4 1 5 2 3	4 1 2 5 3	4 3 5 1 2	4 3 1 5 2
5 4 2 3 1	5 3 1 2 4	5 4 1 2 3	5 4 2 1 3
1 2 3 4 5	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
2 4 1 5 3	2 4 5 1 3	2 3 1 5 4	2 4 5 1 3
3 5 2 1 4	3 1 2 5 4	3 4 5 2 1	3 1 4 5 2
4 1 5 3 2	4 5 1 3 2	4 5 2 1 3	4 5 2 3 1
5 3 4 2 1	5 3 4 2 1	5 1 4 3 2	5 3 1 2 4
1 2 3 4 5	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
2 3 4 5 1	2 5 1 3 4	2 3 4 5 1	2 4 1 5 3
3 1 5 2 4	3 4 2 5 1	3 4 5 1 2	3 1 5 2 4
4 5 1 3 2	4 1 5 2 3	4 5 1 2 3	4 5 2 3 1
5 4 2 1 3	5 3 4 1 2	5 1 2 3 4	5 3 4 1 2
1 2 3 4 5	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
2 3 5 1 4	2 1 4 5 3	2 3 5 1 4	2 4 5 3 1
3 1 4 5 2	3 5 2 1 4	3 1 4 5 2	3 1 2 5 4
4 5 2 3 1	4 3 5 2 1	4 5 1 2 3	4 5 1 2 3
5 4 1 2 3	5 4 1 3 2	5 4 2 3 1	5 3 4 1 2

To avoid having to count how many possibilities there are by hand, use the `nops()` function which gives the number of elements in a list. Based on earlier results, there should be 56.

```
> nops(F);
```

56

>

This shows that this program, although it had to be altered to work with the large number of latin squares for dimension 5, does give all possible latin squares of dimension 5 with the first row and column set.

[back to top](#)

### Part III: Latin squares and group theory

Cayley tables are a huge part of group theory. These tables represent the elements of a group and display the solutions of the operations placed upon

the group. Cayley tables, interestingly enough, have the same properties of latin squares insofar as each element must be used and occurs exactly once in each row and column. Actually, all Cayley tables are latin squares. The question then arises: what is the relationship between latin squares and groups?

Because every group can be written as a Cayley table, and all Cayley tables are latin squares, it is implied that all groups can be represented by a latin square. However, based on the fact that we know how many groups exist of certain orders and the fact that there is such a vast number of latin squares, it seems plausible that every possible latin square (whether or not distinct) may not represent a different group. This mirrors the same idea in group theory, where isomorphic Cayley tables represent the same group. So some latin squares may represent the same groups as other latin squares, but does every latin square represent a group?

Take the groups of order 3. There is only one,  $Z_3$ . (There is only one group of any order  $p$  where  $p$  is prime, namely  $Z_p$ .) Once again limiting the scope to the latin squares produced by the previous programs (where the first row and column are set), it is necessary to determine whether or not the latin square represents a group. In this case, the produced latin square of dimension 3 does represent the group  $Z_3$ .

Moving to the groups of order 4 -- there are two:  $Z_4$  and  $Z_2 \times Z_2$ . However, there are four distinct latin squares produced by latin(4). Which, if any, represent groups? If they all represent groups, it proves the point that some distinct latin squares represent the same group. If any of them does not represent a group, it proves the point that not all latin squares represent groups. Upon further inspection, it is evident that all of these latin squares represent groups of order 4. One of them represents  $Z_2 \times Z_2$ , while the others represent  $Z_4$ .

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 4 & 3 \\ 3 & 4 & 1 & 2 \\ 4 & 3 & 2 & 1 \end{bmatrix}$$
 represents  $Z_2 \times Z_2$  because all of the elements are of order 2.

The other latin squares of dimension 4 have elements of order 4 and order 2, so they represent the cyclic group  $Z_4$ . Again, this investigation is limited to the squares produced with the first row and column set. So, although there may be latin squares of dimension 4 that do not represent groups, the squares

tested under these specifications all represent groups.

Given that 5 is prime, there is only one group of order 5,  $Z_5$ . There are 56 possible latin squares of order 5 under these specifications, which all must represent  $Z_5$  if they are going to represent a group. It would be difficult to go through each one of these possible squares to determine which represent  $Z_5$  and which, if any, do not, but it will suffice to show a few examples to conclude that not all of these squares represent  $Z_5$ .

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 1 & 4 & 5 & 3 \\ 3 & 4 & 5 & 1 & 2 \\ 4 & 5 & 2 & 3 & 1 \\ 5 & 3 & 1 & 2 & 4 \end{bmatrix}$$
 does not represent  $Z_5$  because 2 is not of order 5, and all

elements in  $Z_p$  have to be of order  $p$  to be a Cayley table of  $Z_p$ . Thus, this latin square is not a Cayley table for  $Z_5$ . Similarly,

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 5 & 1 & 4 \\ 3 & 4 & 1 & 5 & 2 \\ 4 & 5 & 2 & 3 & 1 \\ 5 & 1 & 4 & 2 & 3 \end{bmatrix}$$
 does not represent  $Z_5$  because 3 does not have order 5.

These examples prove that not all latin squares actually correspond to groups. However, it is true that all groups have a latin square, for they all can be represented by at least one Cayley table, which are by nature latin squares.

Therefore, all groups can be represented by latin squares, but not all latin squares represent their own unique group.

[back to top](#)

created by: Jamie Woods