Math 7H | Professor: Padraic Bartlett

Lecture 1: Latin Squares and Experimental Design

Week 1 | *UCSB 2015*

I want to start this class with a brief introduction to an object that I spent the bulk of my Ph.D studying: **Latin squares**!

Definition. A **Latin square** of order n is a $n \times n$ array filled with n distinct symbols (by convention $\{1, \dots n\}$), such that no symbol is repeated twice in any row or column.

Example. Here are all of the Latin squares of order 2:

1	2
2	1

2	1
1	2

.

A quick observation we can make is the following:

Proposition. Latin squares exist for all n.

Proof. Behold!

1	2	$\dots$	$n-1$	n
2	3	$\dots$	n	1
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
n	1	$\dots$	$n-2$	$n-1$

□

Given this observation, a natural question to ask might be "How many Latin squares exist of a given order n?" And indeed, this is an excellent question! So excellent, in fact, that it turns out that we have no idea what the answer to it is; indeed, we only know the true number of Latin squares of any given order up to 11.

n	reduced Latin squares of size n[1]	all Latin squares of size n
1	1	1
2	1	2
3	1	12
4	4	576
5	56	161280
6	9408	812851200
7	16942080	61479419904000
8	535281401856	108776032459082956800
9	377597570964258816	5524751496156892842531225600
10	7580721483160132811489280	9982437658213039871725064756920320000
11	5363937773277371298119673540771840	776966836171770144107444346734230682311065600000
12	?	?

Asymptotically, the best we know (and you could show, given a lot of linear algebra tools) that

$$L(n) \sim \left(\frac{n}{e^2}\right)^{n^2}.$$

0.1 Mutually Orthogonal Latin Squares

In these notes, we look at a specific notion related to Latin squares — the concept of "orthogonal" Latin squares! To understand how this works, try solving the following problem:

Question. Take a deck of playing cards, and remove the 16 aces, kings, queens, and jacks from the deck. Can you arrange these cards into a 4×4 array, so that in each column and row, no two cards share the same suit or same face value?

This question should feel similar to the problem of constructing a Latin square: we have an array, and we want to fill it with symbols that are not repeated in any row or column. However, we have the additional constraint that we're actually putting **two** symbols in every cell: one corresponding to a suit, and another corresponding to a face value.

So: if we just look at the face values, we should get a 4×4 Latin square. Similarly, if we ignore the face values and look only at the suits, we should have a different 4×4 Latin square; as well, these two Latin squares ought to have the property that when we superimpose them (i.e. place one on top of the other), each of the resulting possible 16 pairs of symbols occurs exactly once (because we started with 16 distinct cards.)

You can do this! Here is one possible solution:

$A\heartsuit$	$K\diamondsuit$	$Q\spadesuit$	$J\clubsuit$
$K\spadesuit$	$A\clubsuit$	$J\heartsuit$	$Q\diamondsuit$
$Q\clubsuit$	$J\spadesuit$	$A\diamondsuit$	$K\heartsuit$
$J\heartsuit$	$Q\heartsuit$	$K\clubsuit$	$A\spadesuit$

The generalization of this idea is to the concept of **orthogonality**[2] for Latin squares, which we define here:

Definition. A pair of $n \times n$ Latin squares are called **orthogonal** if when we superimpose them (i.e. place one on top of the other), each of the possible n^2 ordered pairs of symbols occur exactly once.

A collection of k $n \times n$ Latin squares is called **mutually orthogonal** if every pair of Latin squares in our collection is orthogonal.

[1] A **reduced** Latin square of size n is a Latin square where the first column and row are both $(1, 2, 3 \ldots n)$. The idea here is that by permuting the rows and columns of any Latin square, you can make it have this "reduced" property. Therefore, in a sense, the only interesting things to count are the number of different reduced squares; this is because from there you can generate any other Latin square by permuting its rows and columns.

[2] This idea has no obvious corresponding geometric context; just think of it as a name for now.

Example. The grid of playing cards we constructed earlier is a pair of 4×4 squares, for the reasons we discussed earlier. To further illustrate the idea, we present a pair of orthogonal 3×3 Latin squares:

1	2	3
2	3	1
3	1	2

,

1	2	3
3	1	2
2	3	1

$\longrightarrow$

$(1,1)$	$(2,2)$	$(3,3)$
$(2,3)$	$(3,1)$	$(1,2)$
$(3,2)$	$(1,3)$	$(2,1)$

Like always, whenever we introduce a mathematical concept in combinatorics, our first instinct should be to attempt to count it! In other words: given an order n, what is the largest collection of mutually orthogonal Latin squares we can find? An upper bound is not too hard to find:

Proposition. For any n, the maximum size of a set of $n \times n$ mutually orthogonal Latin squares is $n - 1$.

Proof. Take any collection $T_1, \dots T_k$ of mutually orthogonal Latin squares. Then notice the following property: if we take any of our Latin squares and permute its symbols (i.e. switch all the 1 and 2's), the new square is still mutually orthogonal to all of the other squares. (Think about this for a bit if you are unpersuaded.)

Using the above observation, notice that we can without any loss of generality assume that the first row of each of our Latin squares is $(1, 2, 3 \dots n)$. Now, take any pair of mutually orthogonal Latin squares from our collection, and look at the symbol in the cell in the first column/second row (i.e. the symbol at $(2, 1)$):

1	2	$\dots$	n
x	$-$	$\dots$	$-$
$\vdots$	$\vdots$	$\ddots$	$\vdots$
$-$	$-$	$\dots$	$-$

,

1	2	$\dots$	n
y	$-$	$\dots$	$-$
$\vdots$	$\vdots$	$\ddots$	$\vdots$
$-$	$-$	$\dots$	$-$

.

We know that neither x nor y can be 1, because both of these squares are Latin squares. As well, we know that they cannot agree, as the first row of the superimposition of these two squares contains the pairs (k, k) , for every $1 \leq k \leq n$. This means that there are at most $n - 1$ squares in our collection $T_1, \dots T_k$, because there are $n - 1$ distinct choices for the cell $(2, 1)$ that are not 1. □

We already know that sometimes $n - 1$ is attainable: in our example above, we found 2 orthogonal Latin squares of order 3. When can we attain this bound?

This (somewhat frustratingly) turns out to be open! That is; we do not know for what values of n this bound is attainable. It's not too hard to prove that it holds for any prime value of n (try it!), but for composite values of n we don't know the answer for almost every possible value.

1 Why We Care

A natural question to ask, before putting more effort into our studies, might be the following: why do we care?

A pure mathematician's answer to this would likely be "because it's interesting!" In mathematics, as a general rule, the practical applications of any field will often only emerge decades after the original work is done. Graph theory — one of the most fundamental tools used in studying big data and the internet today — started off as a almost completely recreational field centered around solving a map-coloring problem. Similarly, most of number theory — the basis for all of today's modern cryptography and security systems — was thought of as "useless" for practical matters by most of its founding fathers. See this mathoverflow thread for some excellent answers!

However, Latin squares have been around long enough to have acquired some obviously useful applications. We'll see some of these next week, in our discussions of error-correcting codes; to finish off today's lecture, however, we'll talk about a few applications to arguably the most applicable of the mathematical sciences: **statistics**!

Specifically, consider the following problem:

Problem. Suppose that you have three varieties of chickens w_1, c_2, c_3. We want to measure which variety of chicken lays the most eggs over the months April, May and June. How can we design an experiment to determine this quickly?

One naïve solution for this problem could be the following:

- Buy three plots of land p_1, p_2, p_3.
- Place chicken flock c_1 in plot p_1, flock c_2 in plot p_2, and flock c_3 in plot p_3.
- Harvest eggs through April, May and June to get yields $y_{A,1}, y_{M,1}, y_{J,1}$ for our first flock c_1, and similarly yields $y_{A,2}, y_{M,2}, y_{J,2}$ and $y_{A,3}, y_{M,3}, y_{J,3}$ for the other two flocks.
- Total our yields, to get numbers $Y_i = Y_{A,i} + Y_{M,i} + Y_{J,i}$ for each of our three chickens. Whichever chicken has the highest total yield is clearly the best chicken breed!

This plan, however, has an obvious flaw: it never switches up the plots of land! This leaves our experiment open to significant flaws and bias; if plot p_1 is (say) exposed to more external noise than the others, or plot p_2 is prone to fox predation, we might have external factors that create bias.

A second solution might be to buy a ton of plots of land in the area, and tend to multiple flocks on many different plots of land. This kind-of works (though you are still exposed to the risk that one given chicken breed variety has "bad luck" with the plots of land it's on,) but it's also remarkably expensive; you have to increase your testing costs by several orders of magnitude, and you're not even guaranteed to have fixed the problem!

A third, still bad solution might be to run tests over **three** years: i.e. to test flock b_i in plot p_i in year 1, in plot p_{i+1} in year 2, and in plot p_{i+2} in year 3. This avoids plot bias: i.e. each flock is tested in each plot. However, this takes **three times as long** to run; in business, this is not a luxury you'll often get!

A much better solution (as you may have guessed) is to use Latin squares! Specifically, consider the following flock rotation schedule:

Plots \ Months	April	May	June
p_1	c_1	c_2	c_3
p_2	c_2	c_3	c_1
p_3	c_3	c_1	c_2

I.e. in April, we have flock c_1 in plot c_1, flock c_2 in plot c_2, and flock c_3 in plot c_3, with flocks rotating at the end of each month. This tests each flock in each plot; therefore, when we sum up our results over the three-month period, we are (in theory) no longer biased by our choice of plots!

We can actually use this idea to do one better. Suppose that we **also** want to test out three different forms of chicken feed f_1, f_2, f_3 at the same time. How can we determine the relative effectiveness of these feeds at the same time as our breed testing, without biasing any of our tests?

The answer here, as you may have guessed, is to use mutually orthogonal Latin squares! In particular, take the two mutually orthogonal 3×3 Latin squares from earlier in lecture:

1	2	3
2	3	1
3	1	2

,

1	2	3
3	1	2
2	3	1

If we use the first of these to plan our flocks and the second of these to plan our feeds, we get the following plan:

c_1	c_2	c_3
c_2	c_3	c_1
c_3	c_1	c_2

,

f_1	f_2	f_3
f_3	f_1	f_2
f_2	f_3	f_1

$\longrightarrow$

Plots\Months	April	May	June
p_1	(c_1, f_1)	(c_2, f_2)	(c_3, f_3)
p_2	(c_2, f_3)	(c_3, f_1)	(c_1, f_2)
p_3	(c_3, f_2)	(c_1, f_3)	(c_2, f_1)

To determine the effectiveness of any breed, we simply sum the yields for that breed's plot over the three months; similarly, to determine the effectiveness of any feed, we sum the yields for that feed's three breeds over the three months it was used in. In both process, each of the "non-relevant" variables that we aren't measuring for all occur equally often, and thus in theory shouldn't interact too much with the data we're trying to measure. This isn't a perfect process; there could be time-sensitive interactions between the feeds and the months, or perhaps certain feeds pair better with certain plots. But it's pretty good, and remarkably efficient at measuring lots of information in a compact and efficient manner! (By contrast, if we wanted to measure each breed against each feed/plot/month possibility, we would need 81 tests; i.e. a ninefold increase in testing expenses!)

So, yes! Latin squares: actually applicable, in addition to being a source of open problems.