Exploring Probabilities in Bingo and Its Variations

Matt Gunn
Advisor: Dr. Bhattacharya
MAA Section Meeting

Outline

- Bingo Overview
- Development of Probability Distribution
- Simulation
- Analysis of a Single Board
- Analysis of Multi-board Games
- Variations of Bingo

Bingo Overview

- Modification of lottery games dating back several centuries
- Commonly played in social organizations, churches, and even casinos

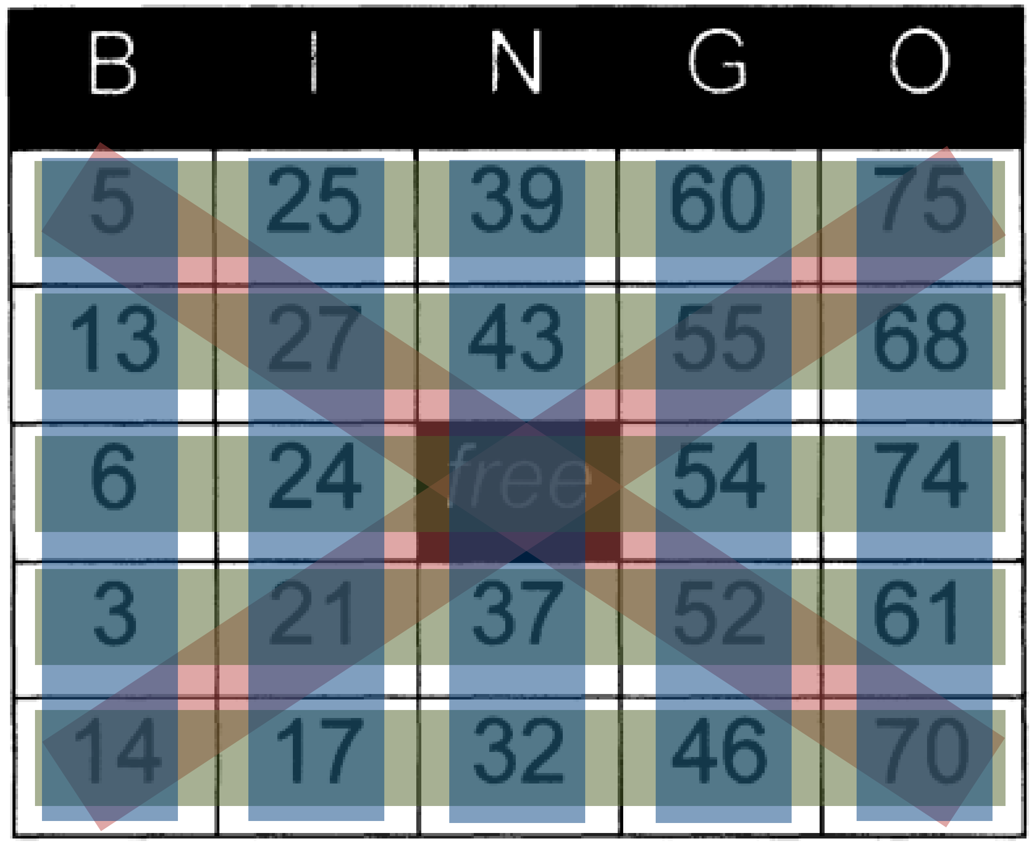
Bingo Overview

- Each player buys a card (or multiple) with a 5x5 gid of squares
- Columns are labeled B, I, N, G, and O.
- The center square is the ‘free space’
- Other squares filled with numbers (Column B: 1-15; Column I: 16-30; Column N: 31-45; Column G: 46-60; Column O: 61-75)
- A caller randomly selects numbers from 1-75 and players mark the appropriate square
- The objective is to be the first to mark an entire row, column, or diagonal

B	I	N	G	O
5	25	39	60	75
13	27	43	55	68
6	24	*free*	54	74
3	21	37	52	61
14	17	32	46	70

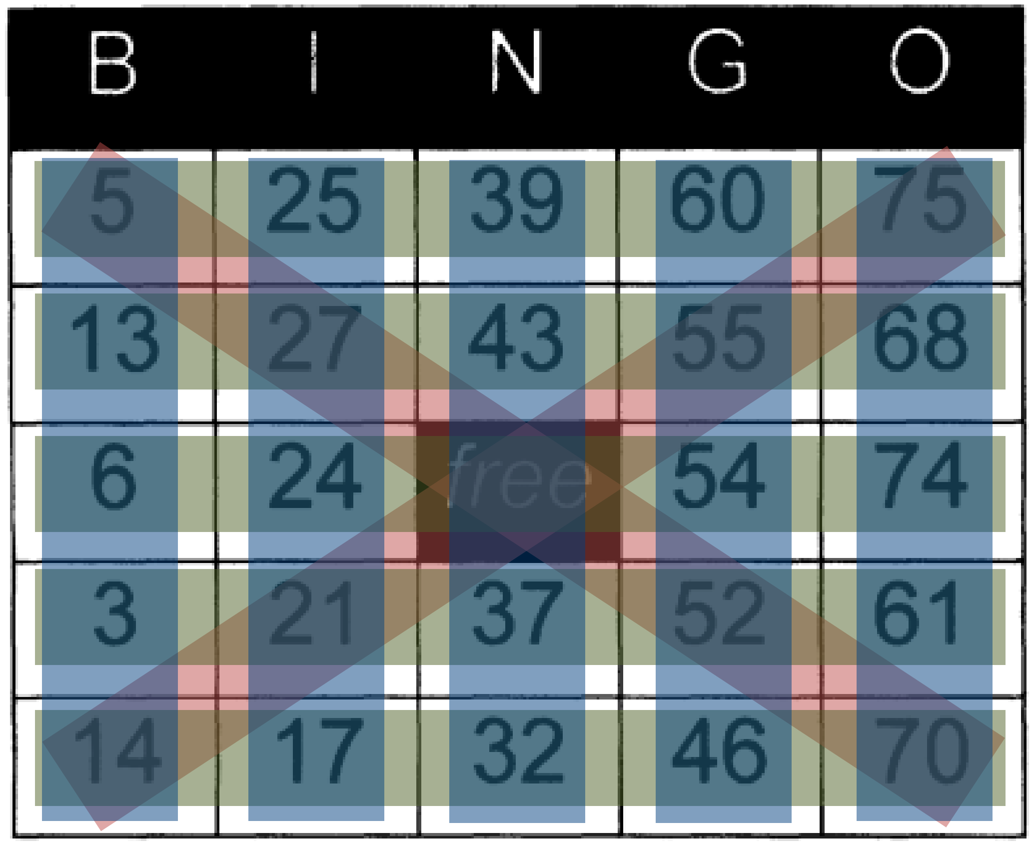
Development of Probability Distribution

- How many different ways are there to get bingo?
- Let’s count



Development of Probability Distribution

- How many different ways are there to get bingo?
- Let’s count
- Let B_i be the probability that the ith bingo is achieved on the kth call
- Thus, the cumulative probability distribution, B, for a bingo in less than k calls is



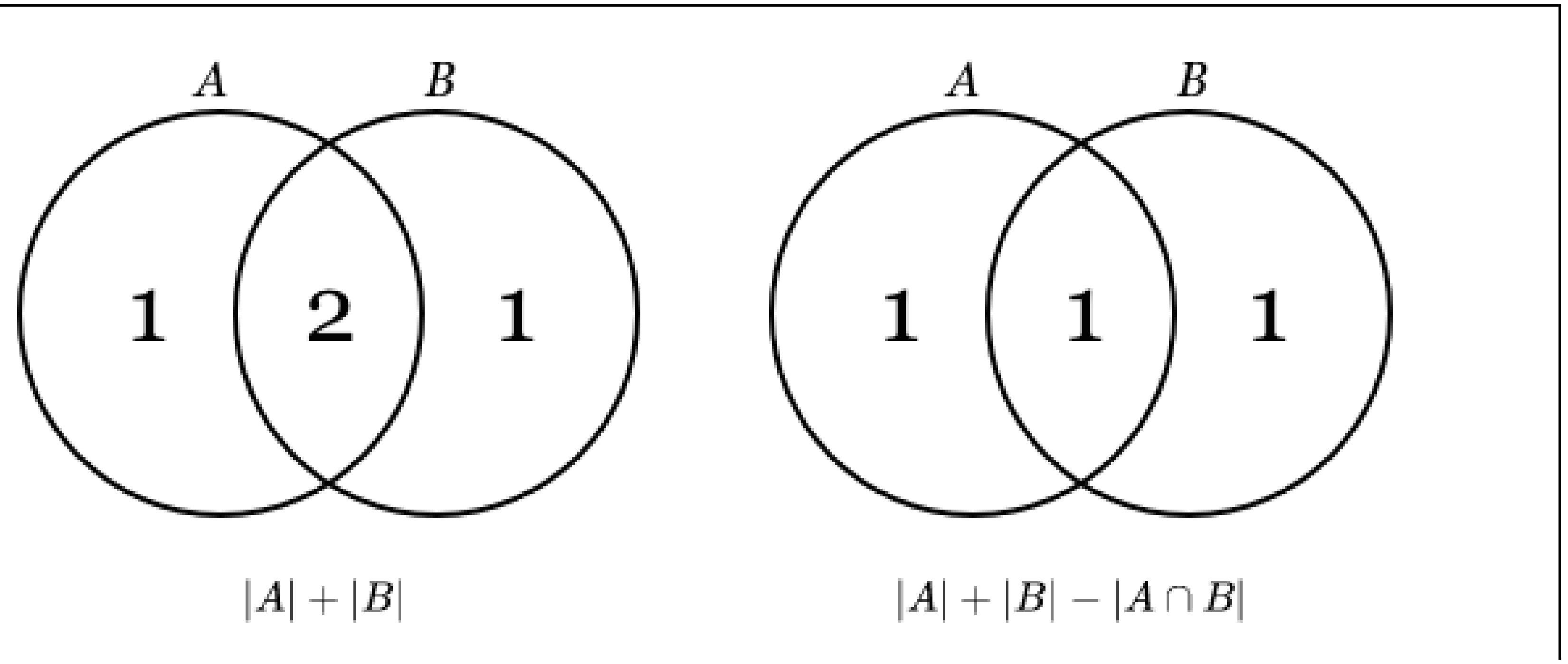
$$P(B \leq k) = P\left(\bigcup_{i=1}^{12} (B_i \leq k)\right)$$

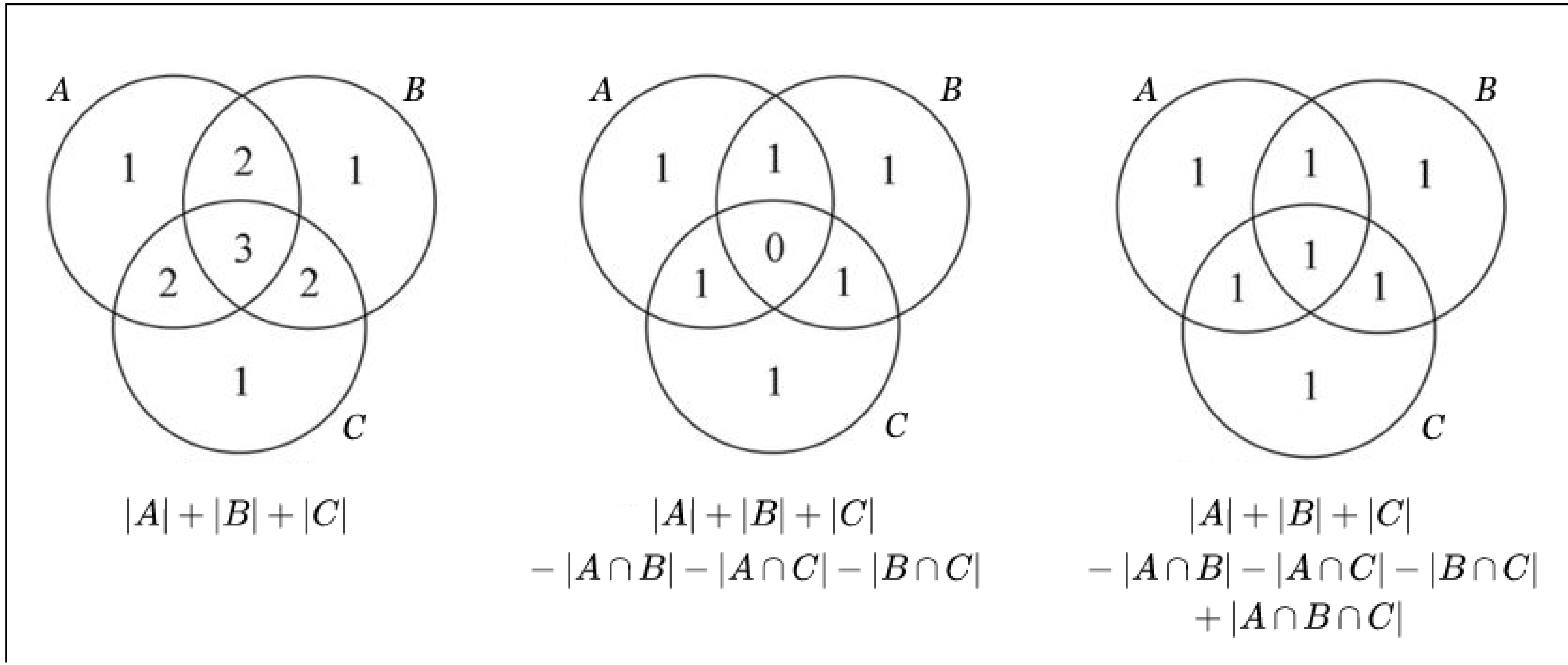
- We can also think of B as the minimum of $B_1, B_2, \ldots, B_{12}$

Inclusion- Exclusion Principle

- The general form for the probability of n events is given by:

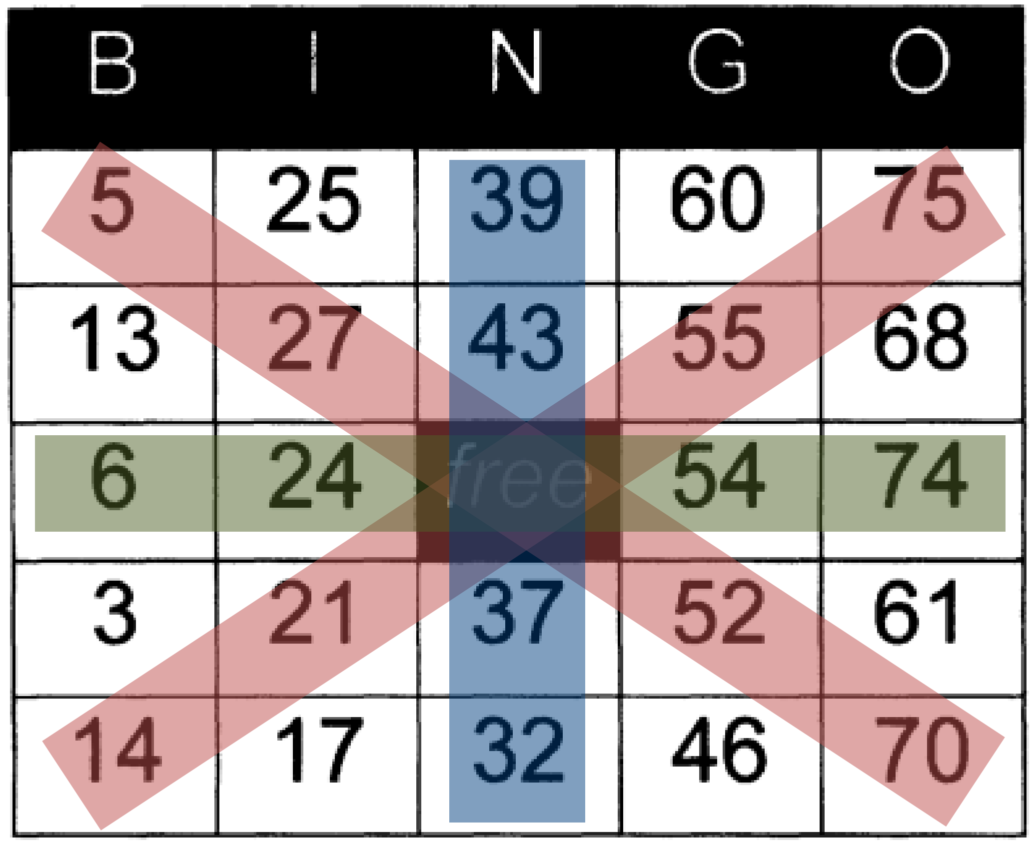
$$P\left(\bigcup_{i=1}^{n} A_i\right) = \sum_i P(A_i) - \sum_{i<j} P(A_i \cap A_j) + \sum_{i<j<k} P(A_i \cap A_j \cap A_k) - \cdots$$
$$+ (-1)^{n+1} P(A_1 \cap A_2 \cdots \cap A_n).$$





Size of Bingo Subsets

- We need to determine the number of possible bingos based on the number of squares
- Let's start with the case of 4 squares covered



Number of squares covered	Number of Bingos (subset size) 1	2	3	4	5	6	7	8	9	10	11	12
4												
5												
6												
7												
8												
9												
10												
11												
12												
13												
14												
15												
16												
17												
18												
19												
20												
21												
22												
23												
24												
Total												

Size of Bingo Subsets

- We need to determine the number of possible bingos based on the number of squares
- Let's start with the case of 4 squares covered

B	I	N	G	O
5	25	39	60	75
13	27	43	55	68
6	24	free	54	74
3	21	37	52	61
14	17	32	46	70

Number of squares covered	Number of Bingos (subset size)											
	1	2	3	4	5	6	7	8	9	10	11	12
4	4											
5												
6												
7												
8												
9												
10												
11												
12												
13												
14												
15												
16												
17												
18												
19												
20												
21												
22												
23												
24												
Total												

Size of Bingo Subsets

- We need to determine the number of possible bingos based on the number of squares
- Let's look at the case of 5 squares

B	I	N	G	O
5	25	39	60	75
13	27	43	55	68
6	24	free	54	74
3	21	37	52	61
14	17	32	46	70

Number of squares covered	Number of Bingos (subset size)											
	1	2	3	4	5	6	7	8	9	10	11	12
4	4											
5												
6												
7												
8												
9												
10												
11												
12												
13												
14												
15												
16												
17												
18												
19												
20												
21												
22												
23												
24												
Total												

Size of Bingo Subsets

- We need to determine the number of possible bingos based on the number of squares
- Let's look at the case of 5 squares

B	I	N	G	O
5	25	39	60	75
13	27	43	55	68
6	24	*free*	54	74
3	21	37	52	61
14	17	32	46	70

Number of squares covered	Number of Bingos (subset size)											
	1	2	3	4	5	6	7	8	9	10	11	12
4	4											
5	8											
6												
7												
8												
9												
10												
11												
12												
13												
14												
15												
16												
17												
18												
19												
20												
21												
22												
23												
24												
Total												

Size of Bingo Subsets

- We need to determine the number of possible bingos based on the number of squares
- Consider the case of 6 squares

B	I	N	G	O
5	25	39	60	75
13	27	43	55	68
6	24	free	54	74
3	21	37	52	61
14	17	32	46	70

Number of squares covered	Number of Bingos (subset size)											
	1	2	3	4	5	6	7	8	9	10	11	12
4	4											
5	8											
6												
7												
8												
9												
10												
11												
12												
13												
14												
15												
16												
17												
18												
19												
20												
21												
22												
23												
24												
Total												

Size of Bingo Subsets

- We need to determine the number of possible bingos based on the number of squares
- Consider the case of 7 squares

B	I	N	G	O
5	25	39	60	75
13	27	43	55	68
6	24	free	54	74
3	21	37	52	61
14	17	32	46	70

Number of squares covered	Number of Bingos (subset size)											
	1	2	3	4	5	6	7	8	9	10	11	12
4	4											
5	8											
6												
7												
8												
9												
10												
11												
12												
13												
14												
15												
16												
17												
18												
19												
20												
21												
22												
23												
24												
Total												

Size of Bingo Subsets

- We need to determine the number of possible bingos based on the number of squares
- Let's look at the case of 8 squares

B	I	N	G	O
5	25	39	60	75
13	27	43	55	68
6	24	free	54	74
3	21	37	52	61
14	17	32	46	70

Number of squares covered	Number of Bingos (subset size)											
	1	2	3	4	5	6	7	8	9	10	11	12
4	4											
5	8											
6												
7												
8												
9												
10												
11												
12												
13												
14												
15												
16												
17												
18												
19												
20												
21												
22												
23												
24												
Total												

Size of Bingo Subsets

- We need to determine the number of possible bingos based on the number of squares
- Let’s look at the case of 8 squares

B	I	N	G	O
5	25	39	60	75
13	27	43	55	68
6	24	free	54	74
3	21	37	52	61
14	17	32	46	70

Number of squares covered	Number of Bingos (subset size)											
	1	2	3	4	5	6	7	8	9	10	11	12
4	4											
5	8											
6												
7												
8												
9												
10												
11												
12												
13												
14												
15												
16												
17												
18												
19												
20												
21												
22												
23												
24												
Total												

Size of Bingo Subsets

- We need to determine the number of possible bingos based on the number of squares
- Let’s look at the case of 8 squares

B	I	N	G	O
5	25	39	60	75
13	27	43	55	68
6	24	free	54	74
3	21	37	52	61
14	17	32	46	70

Number of squares covered	Number of Bingos (subset size)											
	1	2	3	4	5	6	7	8	9	10	11	12
4	4											
5	8											
6												
7												
8												
9												
10												
11												
12												
13												
14												
15												
16												
17												
18												
19												
20												
21												
22												
23												
24												
Total												

Size of Bingo Subsets

- We need to determine the number of possible bingos based on the number of squares
- Let's look at the case of 8 squares

B	I	N	G	O
5	25	39	60	75
13	27	43	55	68
6	24	free	54	74
3	21	37	52	61
14	17	32	46	70

Number of squares covered	Number of Bingos (subset size)											
	1	2	3	4	5	6	7	8	9	10	11	12
4	4											
5	8											
6												
7												
8												
9												
10												
11												
12												
13												
14												
15												
16												
17												
18												
19												
20												
21												
22												
23												
24												
Total												

Size of Bingo Subsets

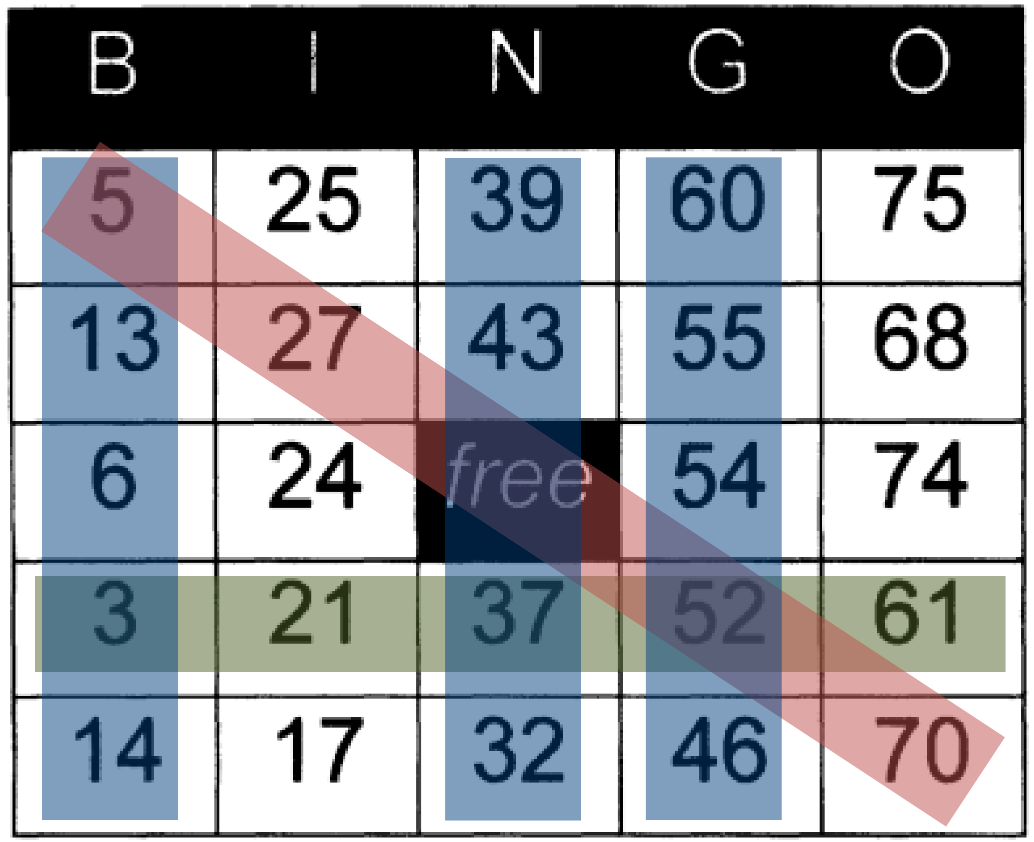
- We need to determine the number of possible bingos based on the number of squares
- Let's look at the case of 8 squares

B	I	N	G	O
5	25	39	60	75
13	27	43	55	68
6	24	free	54	74
3	21	37	52	61
14	17	32	46	70

Number of squares covered	Number of Bingos (subset size)											
	1	2	3	4	5	6	7	8	9	10	11	12
4	4											
5	8											
6												
7												
8												
9												
10												
11												
12												
13												
14												
15												
16												
17												
18												
19												
20												
21												
22												
23												
24												
Total												

Size of Bingo Subsets

- We need to determine the number of possible bingos based on the number of squares
- Standard counting method given certain bingos
- 4 squares, 5 squares, subtract intersections
- 12-digit binary representation for bingos



Number of squares covered	Number of Bingos (subset size)											
	1	2	3	4	5	6	7	8	9	10	11	12
4	4											
5	8											
6												
7												
8												
9												
10												
11												
12												
13												
14												
15												
16												
17												
18												
19												
20												
21												
22												
23												
24												
Total												

Size of Bingo Subsets

- We need to determine the number of possible bingos based on the number of squares
- Continue this pattern for the rest of the table
- The probability of completing any set of n squares in ≤ 75:

B	I	N	G	O
5	25			
13	27			
6	24			
3	21	37	52	61
14	17	32	46	70

$$P(S_n \leq k) = \frac{\binom{75-n}{k-n}}{\binom{75}{k}}.$$

Number of squares covered	Number of Bingos (subset size) 1	2	3	4	5	6	7	8	9	10	11	12
4	4											
5	8											
6												
7												
8		30										
9		24										
10		12										
11			48									
12			104									
13			48	8								
14			12	148								
15			8	152	8							
16				145	120	2						
17				32	232	8						
18					312	136	4					
19				8	48	304	24					
20				2	62	256	182	10				
21					8	192	264	56				
22						12	268	228	36			
23							8	128	96	16		
24					2	14	42	73	88	50	12	1
Total	12	66	220	495	792	924	792	495	220	66	12	1

Probability Distribution

- Our probability distribution then becomes

$$P(B \le k) = P\left(\bigcup_{i=1}^{12} (B_i \le k)\right)$$

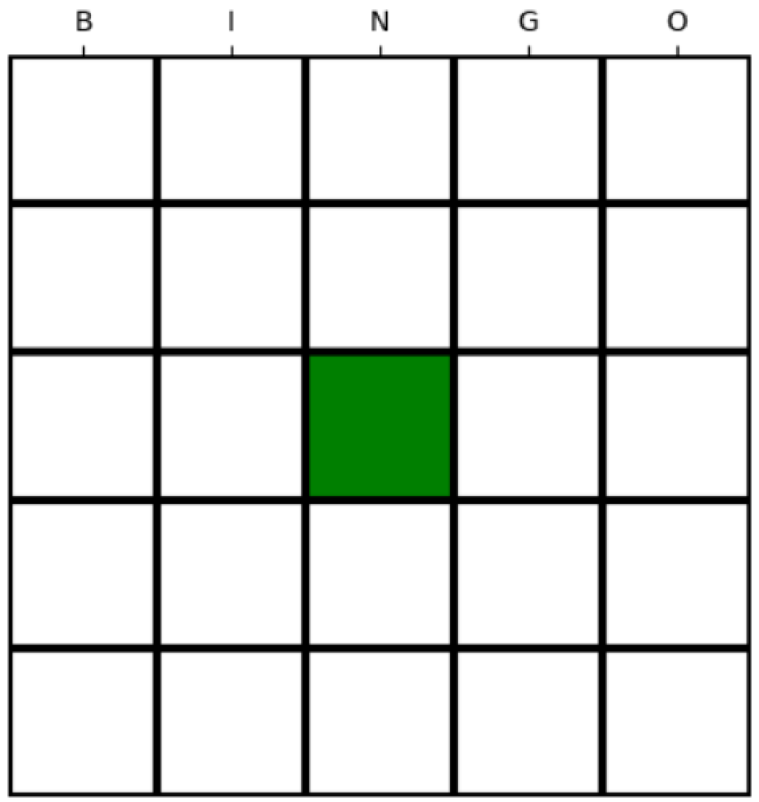
- a_{ni} represents the entry from the table corresponding to n covered squares in the ith column
- From this equation we can calculate a probability distribution for the number of calls (k) to complete a bingo

k	5	10	15	20	25	30	35	40	45	50	55	60	65
$P(B \le k)$	.00002	.0008	.0059	.0229	.0640	.1435	.2719	.4456	.6401	.8144	.9322	.9859	.9990

Number of squares covered	Number of Bingos (subset size)											
	1	2	3	4	5	6	7	8	9	10	11	12
4	4											
5	8											
6												
7												
8		30										
9		24										
10		12										
11			48									
12			104									
13			48	8								
14			12	148								
15			8	152	8							
16				145	120	2						
17				32	232	8						
18					312	136	4					
19				8	48	304	24					
20				2	62	256	182	10				
21					8	192	264	56				
22						12	268	228	36			
23							8	128	96	16		
24					2	14	42	73	88	50	12	1
Total	12	66	220	495	792	924	792	495	220	66	12	1

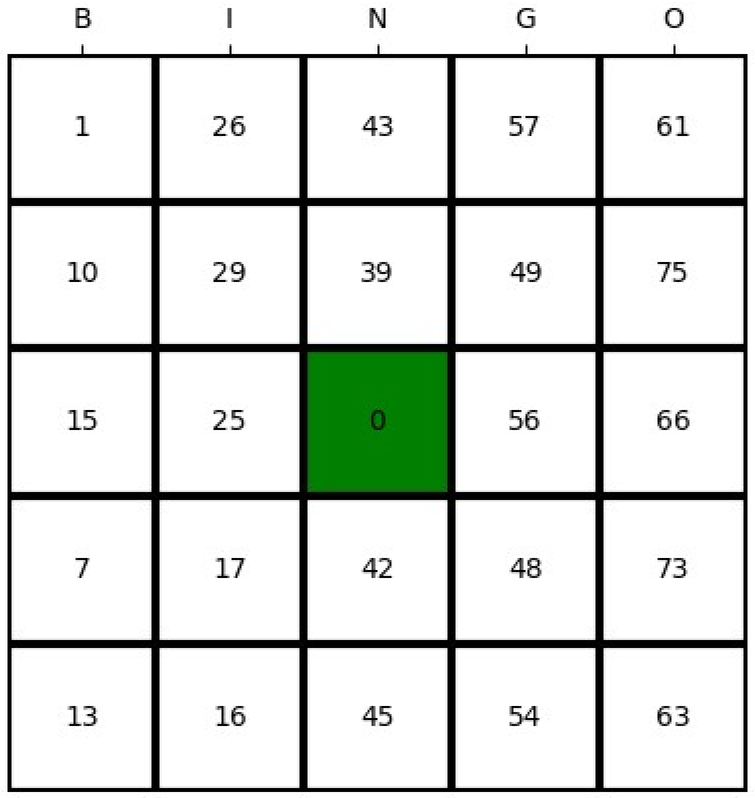
Python Simulation

- Generate a blank 5x5 board



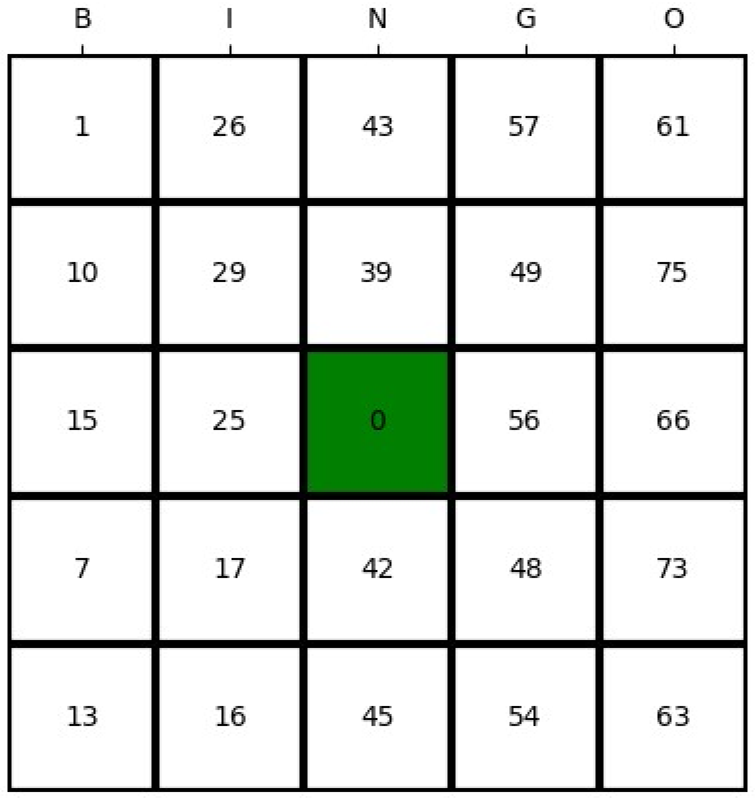
Python Simulation

- Generate a blank 5x5 board
- Randomly fill the board with the right numbers



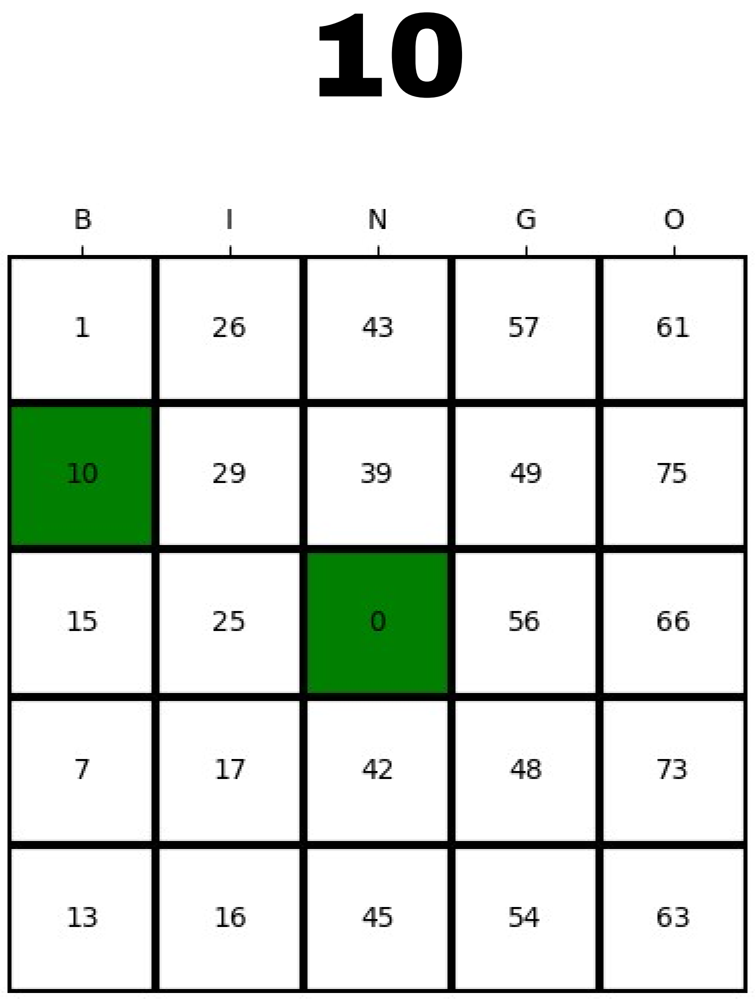
Python Simulation

- Generate a blank 5x5 board
- Randomly fill the board with the right numbers
- Generate a random sequence of unique numbers 1-75 to serve as the call sequence



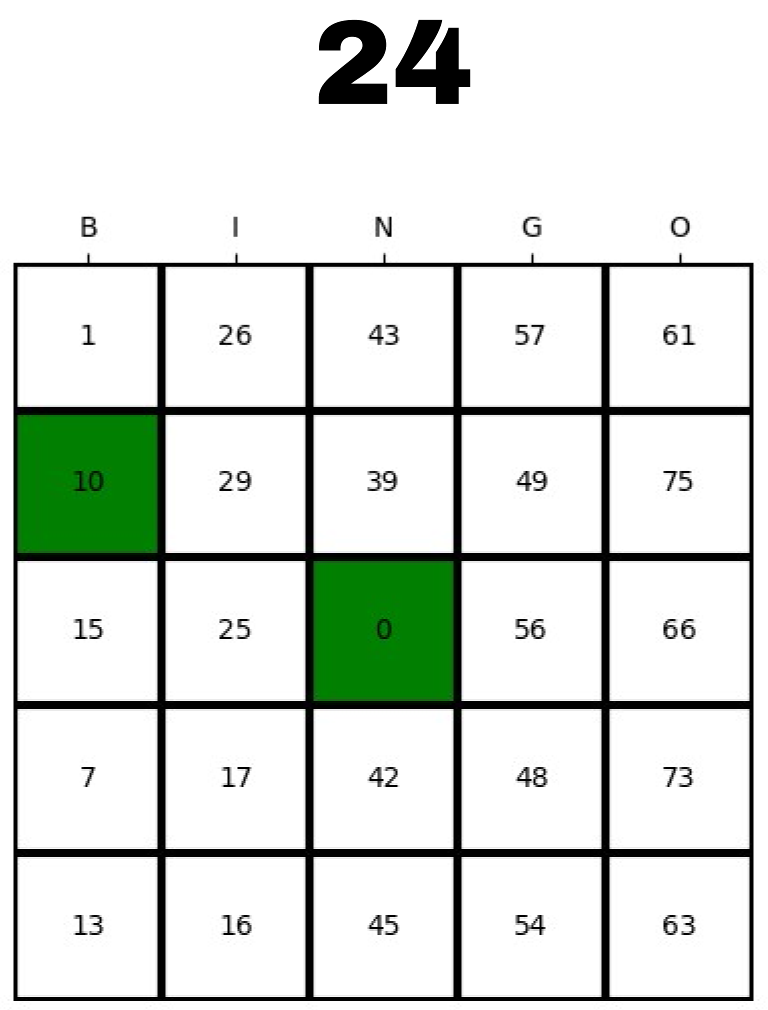
Python Simulation

- Generate a blank 5x5 board
- Randomly fill the board with the right numbers
- Generate a random sequence of unique numbers 1-75 to serve as the call sequence
- Call a number, and mark the board if there is match



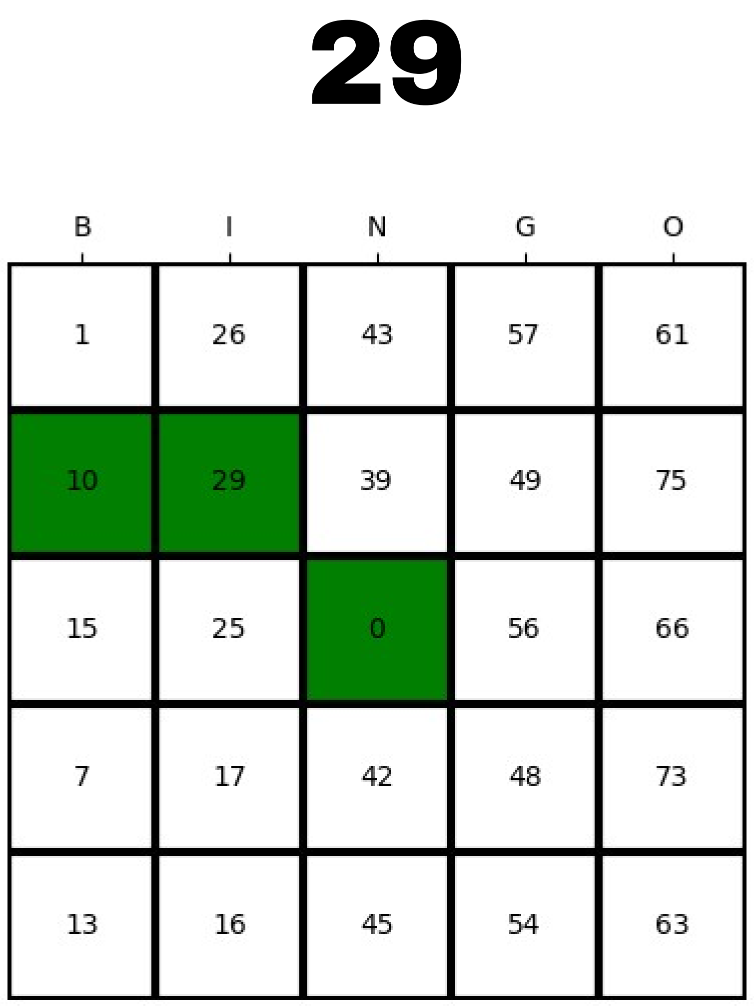
Python Simulation

- Generate a blank 5x5 board
- Randomly fill the board with the right numbers
- Generate a random sequence of unique numbers 1-75 to serve as the call sequence
- Call a number, and mark the board if there is match
- Continue calling and marking numbers until there is a bingo



Python Simulation

- Generate a blank 5x5 board
- Randomly fill the board with the right numbers
- Generate a random sequence of unique numbers 1-75 to serve as the call sequence
- Call a number, and mark the board if there is match
- Continue calling and marking numbers until there is a bingo



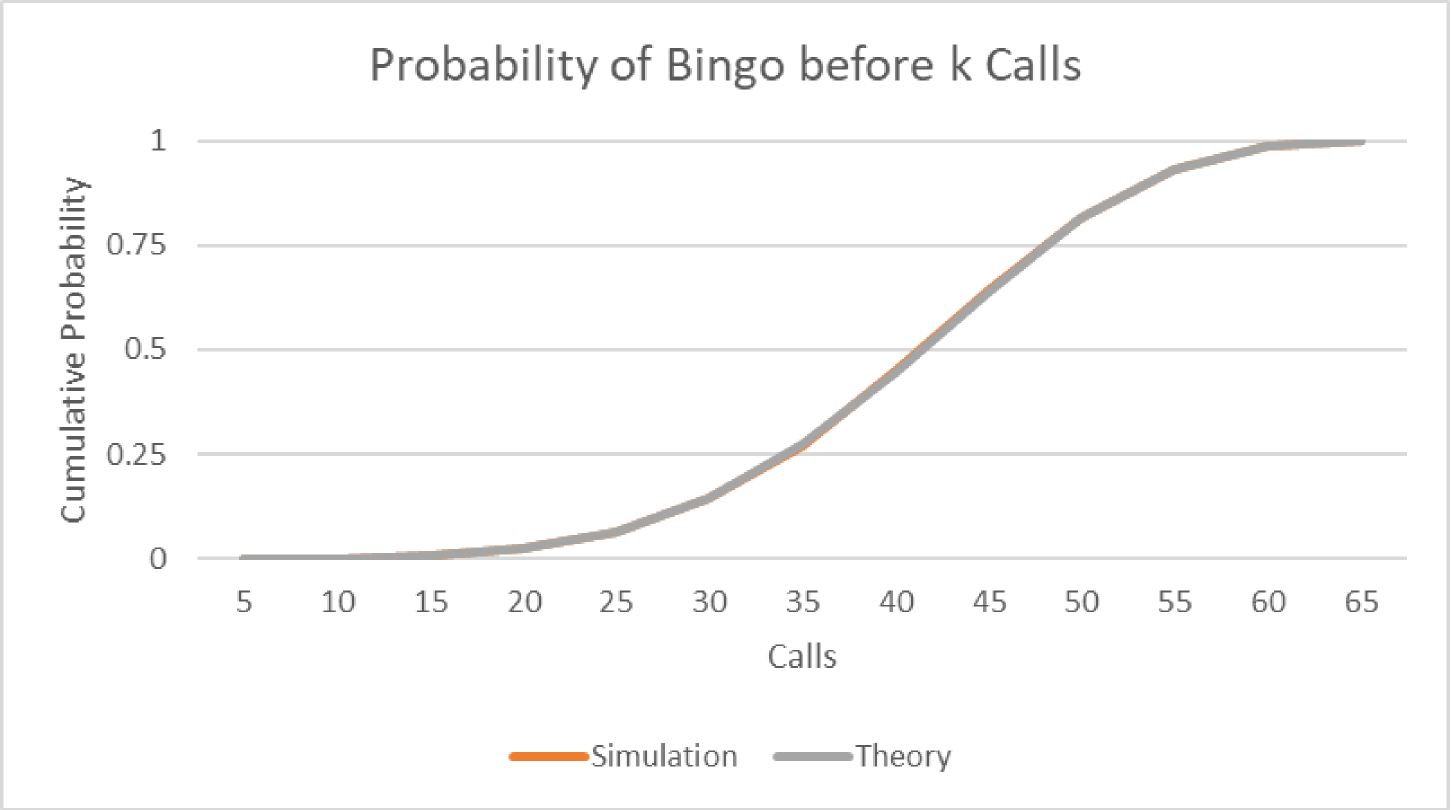
Bingo!

1	26	43	57	61
10	29	39	49	75
15	25	0	56	66
7	17	42	48	73
13	16	45	54	63

Python Simulation

- We will play 100,000 games of bingo to estimate a probability distribution

Calls	*Simulation Probability*	*Theory Probability*	*Percent Error*
5	0.00002	0.00002	0.00%
10	0.00082	0.0008	2.50%
15	0.0058	0.0059	1.69%
20	0.02282	0.0229	0.35%
25	0.06386	0.064	0.22%
30	0.14389	0.1435	0.27%
35	0.27066	0.2719	0.46%
40	0.44817	0.4456	0.58%
45	0.64261	0.6401	0.39%
50	0.81485	0.8144	0.06%
55	0.93307	0.9322	0.09%
60	0.98597	0.9859	0.01%
65	0.99903	0.999	0.00%



Multiple Boards

- In practice, bingo is normally played with many cards involved
- If we assume independence of m cards, then the probability of no bingo after k calls is

$$[1 - P(B \leq k)]^m \, .$$

- Thus, the probability that the first bingo $B_{(1)}$ will occur in at most k calls is

$$P(B_{(1)} \leq k) = 1 - [1 - P(B \leq k)]^m$$

- However, the assumption of independence is not accurate, which makes the analysis substantially more difficult because of conditional probabilities
- This makes a simulation helpful to approximate the cumulative probability distribution

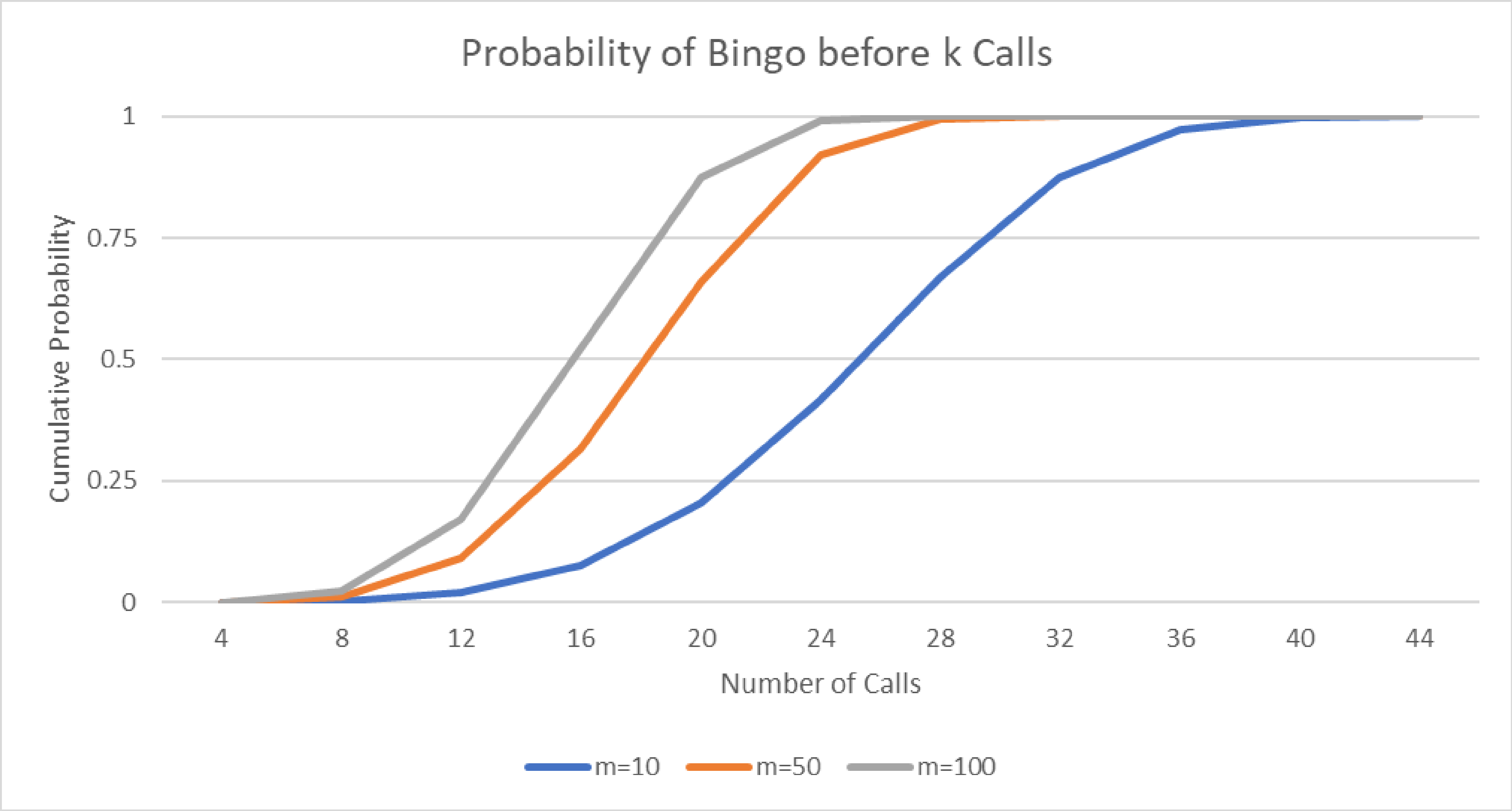
Multiple Board Simulation

- Generate m blank 5x5 boards
- Randomly fill each board with the right numbers
- Generate a random sequence of unique numbers 1-75 to serve as the call sequence
- Call a number, and mark each board if there is match
- Continue calling and marking numbers until there is a bingo

Multiple Board Simulation

- We will play 100,000 games of bingo to estimate a probability distribution

	4	8	12	16	20	24	28	32	36	40	44
m=10	0.0000	0.0024	0.0198	0.0766	0.2059	0.4167	0.6691	0.8732	0.9725	0.9976	1.0000
m=50	0.0000	0.0121	0.0918	0.3172	0.6611	0.9212	0.9945	0.9999	1.0000	1.0000	1.0000
m=100	0.0000	0.0246	0.1704	0.5213	0.8745	0.9916	0.9999	1.0000	1.0000	1.0000	1.0000



Blackout

- All squares must be covered

Calls	Frequency	Cumulative %
51	4	0.00%
53	1	0.01%
55	7	0.01%
57	25	0.04%
59	65	0.10%
61	153	0.26%
63	369	0.62%
65	937	1.56%
67	2205	3.77%
69	5100	8.87%
71	11459	20.33%
73	25443	45.77%
75	54232	100.00%

B	I	N	G	O
5	25	39	60	75
13	27	43	55	68
6	24	free	54	74
3	21	37	52	61
14	17	32	46	70

4 corners & Postage Stamp

B	I	N	G	O
5	25	39	60	75
13	27	43	55	68
6	24	free	54	74
3	21	37	52	61
14	17	32	46	70

Calls	*4 Corners*	*Postage Stamp*
5	0.0000	0.0000
10	0.0002	0.0002
15	0.0010	0.0010
20	0.0039	0.0039
25	0.0099	0.0106
30	0.0218	0.0221
35	0.0421	0.0433
40	0.0744	0.0749
45	0.1217	0.1223
50	0.1892	0.1893
55	0.2814	0.2812
60	0.4009	0.4009
65	0.5551	0.5551
70	0.7535	0.7532
75	1.0000	1.0000

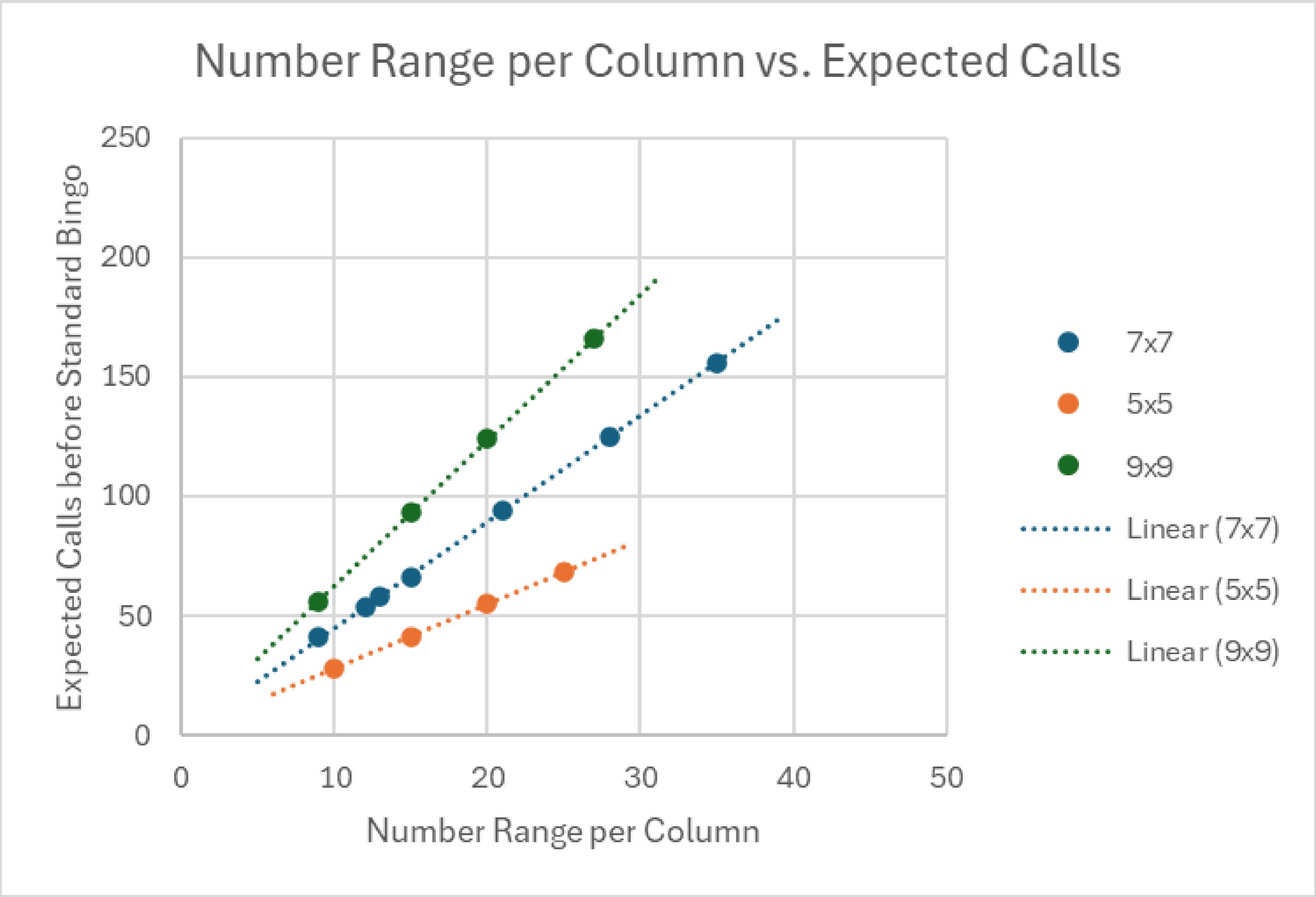
B	I	N	G	O
5	25	39	60	75
13	27	43	55	68
6	24	free	54	74
3	21	37	52	61
14	17	32	46	70

Larger Bingo Boards

X	L	B	I	N	G	O
13	23	39	57	67	82	102
10	24	43	47	63	76	103
14	30	38	55	73	78	104
8	21	32	0	75	86	95
1	29	33	48	62	89	101
5	19	41	60	65	85	91
2	22	40	58	74	84	92

V	A	S	T	B	I	N	G	O
19	30	63	107	119	147	163	206	243
26	48	75	92	132	157	174	210	236
8	40	74	101	117	160	186	215	225
16	32	80	93	127	142	183	203	235
17	42	71	98	0	151	168	208	218
13	49	61	95	131	149	177	202	228
7	31	57	99	122	150	176	207	239
22	34	62	94	125	139	179	205	221
9	39	78	86	111	137	182	212	226

Larger Bingo Boards



QUESTIONS?

References

Agard, David B., and Michael W. Shackleford. “A new look at the probabilities in Bingo.” *The College Mathematics Journal*, vol. 33, no. 4, 2002, pp. 301–305, https://doi.org/10.1080/07468342.2002.11921957.

Mercer, Joseph O. “Some surprising probabilities from bingo.” *The Mathematics Teacher*, vol. 86, no. 9, 1993, pp. 726–731, https://doi.org/10.5951/mt.86.9.0726.

“Principle of Inclusion and Exclusion (PIE).” Brilliant Math & Science Wiki, brilliant.org/wiki/principle-of-inclusion-and-exclusion-pie/. Accessed 30 Nov. 2023.

“We’ve Always Got Your Number at Red Rock Bingo.” *Redrockresort.Com*, www.redrockresort.com/play/bingo/. Accessed 30 Nov. 2023.

Bingo Board

B	I	N	G	O
5	25	39	60	75
13	27	43	55	68
6	24	free	54	74
3	21	37	52	61
14	17	32	46	70

Code

```
import numpy as np
import seaborn as sns

def playBingo():
  # Make a Bingo Card
  # Make an empty 5x5
  card=np.zeros((5,5))
  # Loop over rows and columns
  for i in range(0,5):
    for j in range(0,5):
      # Skip free space (position 2,2)
      if not(i==2 and j==2):
        # Generate a random number in the right interval
        num=np.random.randint((j*15)+1,(j*15)+16)
        # Keep generating numbers until we find one that is not in the Bingo board already
        while num in card:
          num=np.random.randint((j*15)+1,(j*15)+16)
        # Add the unique number to the board
        card[i,j]=num
```

Code

```
# Generate the call order for the numbers
# Make an array of the integers 1-75
uncalledNums=np.linspace(1,75,75)
uncalledNums=list(uncalledNums)
callOrder=np.zeros(75)
for i in range(0,75):
  # Randomly pick an index of in uncalledNums
  idx=np.random.randint(0,75-i)
  callOrder[i]=uncalledNums.pop(idx)
```

Code

```python
# Call each number and mark the board untill bingo
bingo= False
calls=0
while not bingo and calls < 75:
  num=callOrder[calls]
  idx=np.where(card==num)
  card[idx]=0
  # Determine if there is Bingo
  if calls>=4: # at least 4 calls are required for any Bingo
    # Check for diagonal Bingo
    if card[0,0]+card[1,1]+card[2,2]+card[3,3]+card[4,4]==0:
      bingo=True
    if card[0,4]+card[1,3]+card[2,2]+card[3,1]+card[4,0]==0:
      bingo=True
    # Check is the sum of a column or row is zero
    i=0 # index for column/row
    while not bingo and i<5: # Stop checking if bingo or checked all indexes
      if sum(card[i,:])==0 or sum(card[:,i])==0: # Check sum of ith column and row
        bingo=True
      i=i+1

  calls=calls+1

return calls
```

Code

```python
def main():
  n=100000
  results=np.zeros(n)
  for i in range(0,n):
    results[i]=playBingo()
  #print(results)
  np.savetxt("results.csv", results, delimiter=",")
main()
```