

EE/CpE 345

Modeling and Simulation

Fall 2003

Class 7

Random-Number Generation

- Properties of Random Numbers
- Generating Pseudo-Random Numbers
- Techniques for Generating Random Numbers
- Testing for Randomness

Uses for Random Numbers in Simulations

- Event scheduling
 - Customer arrivals
 - Service times
 - Failures
- Noise generation
- Any non-deterministic process

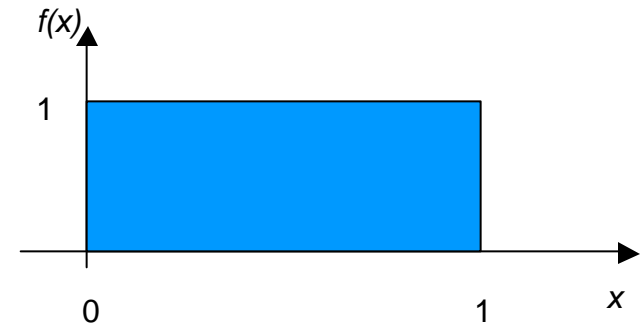
Properties for Uniformly Distributed Random Numbers

- Uniformity

$$f(x) = \begin{cases} 1, & 0 \leq x \leq 1 \\ 0, & \text{otherwise} \end{cases}$$

$$E(R) = \int_0^1 x dx = x^2 \Big|_0^1 = \frac{1}{2}$$

$$V(R) = \int_0^1 x^2 dx - [E(R)]^2 = \frac{x^3}{3} \Big|_0^1 - \left(\frac{1}{2}\right)^2 = \frac{1}{3} - \frac{1}{4} = \frac{1}{12}$$



- Independence

$$\forall (a, b, a \leq b) \in (0, 1), \quad P(a \leq X \leq b) = \frac{1}{b-a}$$

$$\forall (X_1, X_2, \dots, X_{i-1}), \forall (a, b, a \leq b) \in (0, 1), \quad P(a \leq X_i \leq b) = \frac{1}{b-a}$$

Generating Truly Random Numbers

- Process would need to be based on some known natural random process:
 - Decay of radioactive material
 - Noise generated in electronic circuit with no deterministic interference (e.g., hum)
 - Relative jitter of two independent clock sources
- These are impractical to generate for all but the most stringent requirements for true randomness
- Usually we settle for Pseudo-random numbers in simulations

Generating Pseudo-Random Numbers

- Generated by some known (e.g., deterministic) process
- Pseudo-random numbers *model* random numbers for simulation purposes
- Potential issues:
 - Non-uniformity
 - Discrete valued, not continuously valued
 - Inaccurate mean
 - Inaccurate variance
 - Dependence
 - Autocorrelation between numbers
 - Runs of numbers with skewed values, with respect to previous numbers or mean value

Selecting a Pseudo-Random Number Generating Routine

- Requirements:
 - Speed - a large number of random numbers will be needed for a typical simulation.
 - Portability - between machines (with different numeric precision, word order), between different languages
 - Long cycle-time - All pseudo-random generators repeat.
 - Repeatability - You might want to rerun same simulation with same conditions
 - Statistically representative of true random numbers
 - Uniformity
 - Independence
 - Other statistical tests
- “Randomness is in the eye of the beholder”
 - It is easy to generate numbers that *look* random
 - It is hard to generate numbers that pass rigorous statistical tests and meet the requirements

Pseudo-Random Number Generation Techniques

- Linear Congruential Method

$$X_{i+1} = (aX_i + c) \bmod m, \quad i = 0, 1, 2, \dots$$

X_0 - seed

a - constant multiplier

c - increment

m - modulus

- $c=0$: *multiplicative congruential method*, otherwise *mixed congruential method*
- Choice of a , c , and m have dramatic effect on statistical properties of resulting numbers

Example 7.1

Example 7.1					
Linear Congruential Pseudo Random Number Generator					
X0= 27					
a= 17					
c= 43					
m= 100					
$B_{10} = \text{MOD}(B_9 * a + c, m)$					
index	x(index)	normalized			
0	27	0.27			
1	2	0.02			
2	77	0.77			
3	52	0.52			

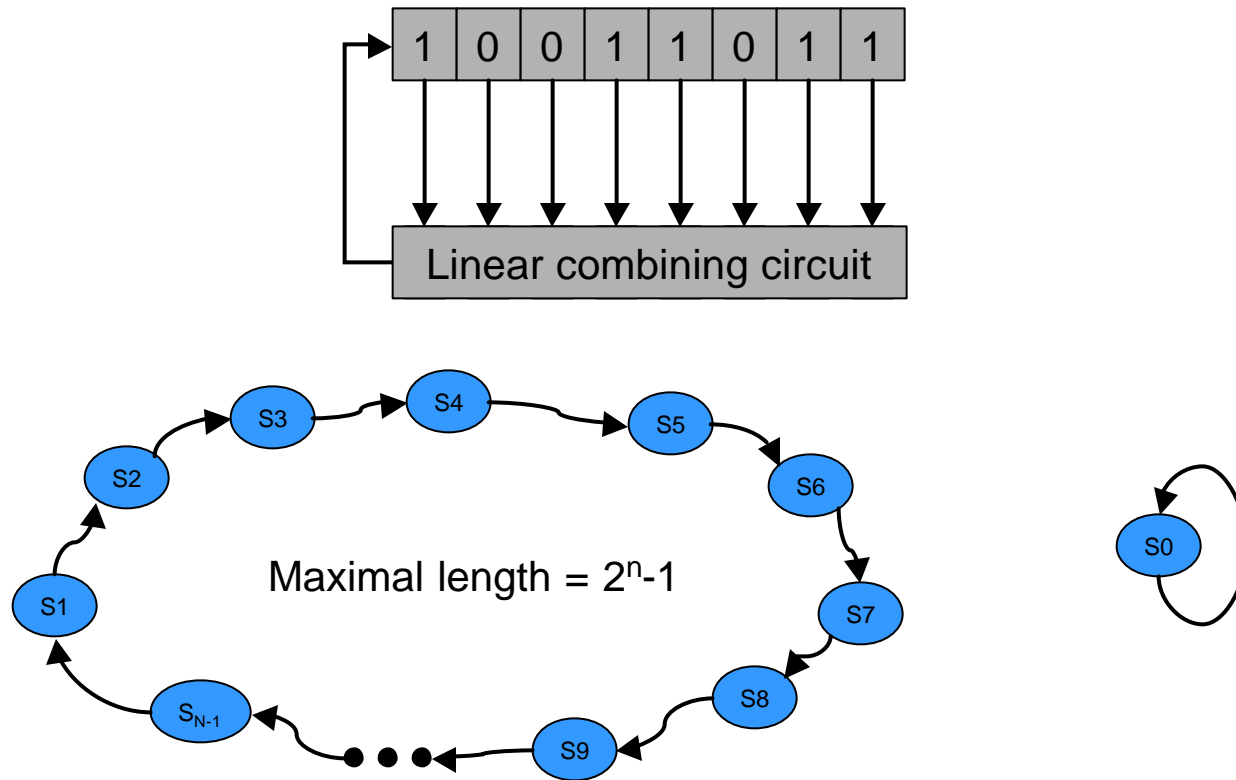
Example 7.1 - the Real Story

Example 7.1					
Linear Congruential Pseudo Random Number Generator					
	X0=	27			
	a=	17			
	c=	43			
	m=	100			
	B10=+MOD(B9*_a+_c,_m)				
index	x(index)	normalized			
0	27	0.27			
1	2	0.02			
2	77	0.77			
3	52	0.52			
4	27	0.27			
5	2	0.02			
6	77	0.77			
7	52	0.52			
8	27	0.27			
9	2	0.02			
10	77	0.77			
11	52	0.52			
12	27	0.27			
13	2	0.02			
14	77	0.77			
15	52	0.52			
16	27	0.27			
17	2	0.02			
18	77	0.77			
19	52	0.52			
20	27	0.27			

Example 7.1 - With a Different X_0

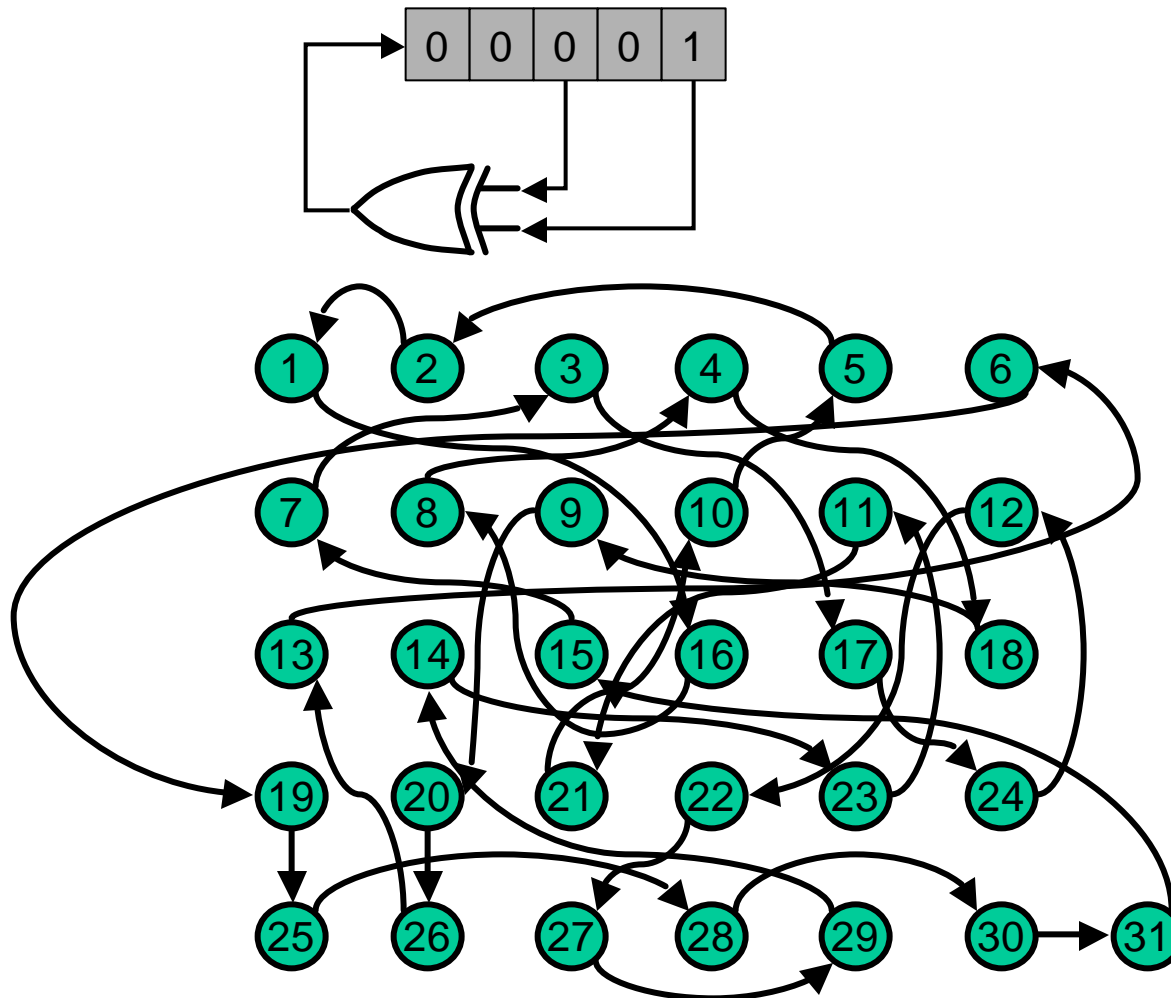
Example 7.1					
Linear Congruential Pseudo Random Number Generator					
	$X_0 =$	13			
	$a =$	17			
	$c =$	43			
	$m =$	100			
	$B_{i+1} = \text{MOD}(B_i * a + c, m)$				
index	$x(\text{index})$	normalized			
0	13	0.13			
1	64	0.64			
2	31	0.31			
3	70	0.7			
4	33	0.33			
5	4	0.04			
6	11	0.11			
7	30	0.3			
8	53	0.53			
9	44	0.44			
10	91	0.91			
11	90	0.9			
12	73	0.73			
13	84	0.84			
14	71	0.71			
15	50	0.5			
16	93	0.93			
17	24	0.24			
18	51	0.51			
19	10	0.1			
20	13	0.13			

Linear Congruential Methods and Maximal Length Shift Registers



- With $m=2^N$, and N =computer word size, modular reduction is free

Maximal Length Shift Registers Example



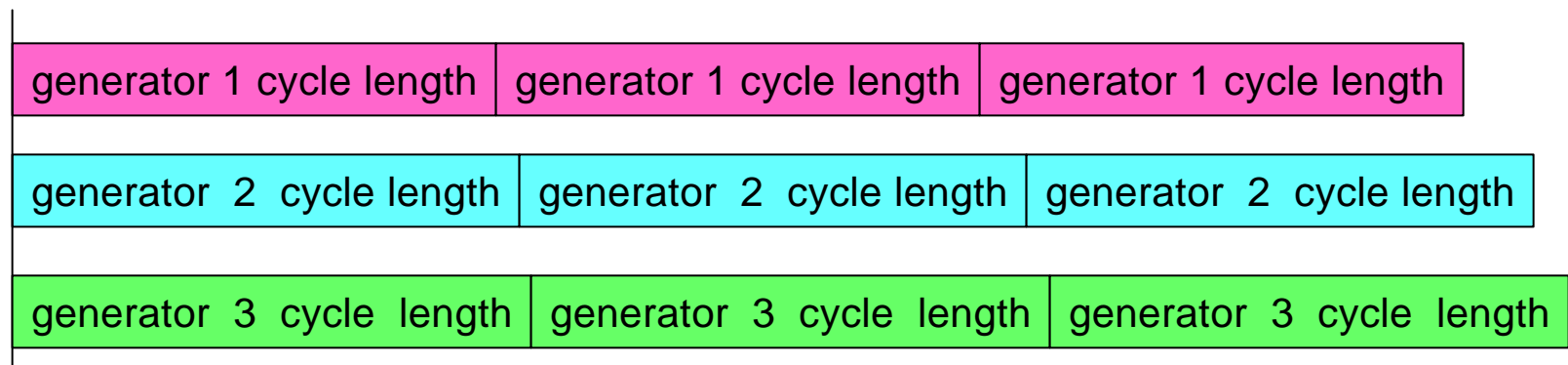
Example 7.4

Example 7.4		
Linear Congruential Pseudo Random Number Generator		
X0=	123457	
a=	16807	
c=	0	
m=	2147483647	
	$B_{i+1} = \text{MOD}(B_i * a + c, m)$	
index	x(index)	normalized
0	123457	5.74891E-05
1	2074941799	0.966220069
2	559872160	0.260710791
3	1645535613	0.766262232
4	1222641625	0.569336873
5	1814256879	0.844829194
6	95061600	0.044266507
7	2119961479	0.987183991
8	1291390176	0.601350412
9	1924951450	0.89637537
10	817878095	0.380854167
11	34318218	0.015980666
12	1260672530	0.587046393
13	1049550408	0.488734994
14	363030798	0.169049389
15	457580859	0.213077692
16	422557306	0.196768579
17	192221313	0.089510024
18	848202503	0.394975069
19	743019135	0.345995247
20	305194640	0.142117329

- normalization by $m+1$, but $m/(m+1) \sim 1$

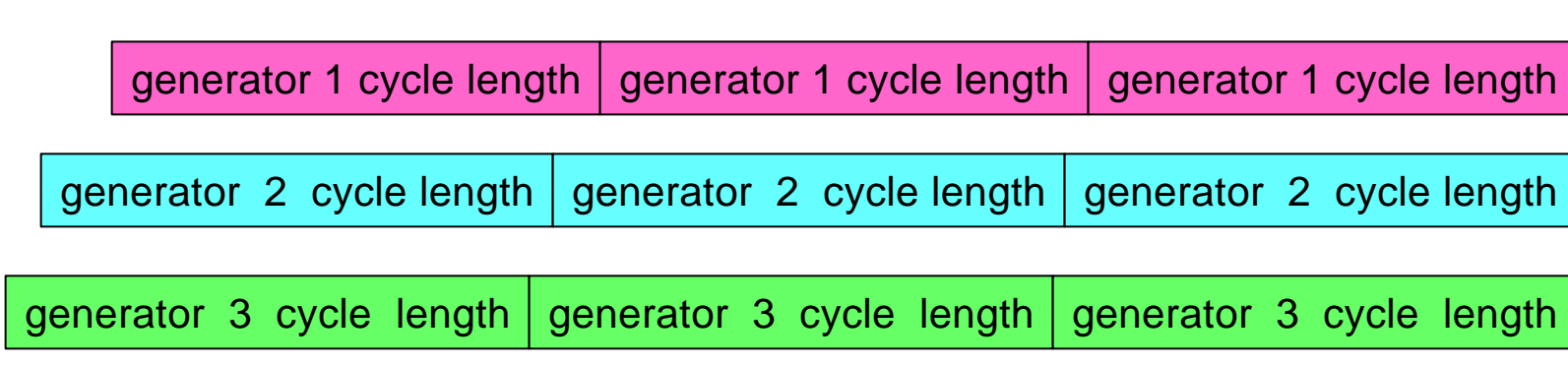
Combining Linear Congruential Generators

- Example 7.4 has a period of $2^{31}-1 \sim 2 \times 10^9$
- When mainframe and minicomputers were less than 1 MIP machines, simulations requiring billions of random numbers were unheard of. The generator never cycled during one simulation
- With PCs capable of 100's of MIPS, simulations are getting bigger.
- 10^9 cycle times are too short.
- Choose multiple generators with *relatively prime* cycle lengths:

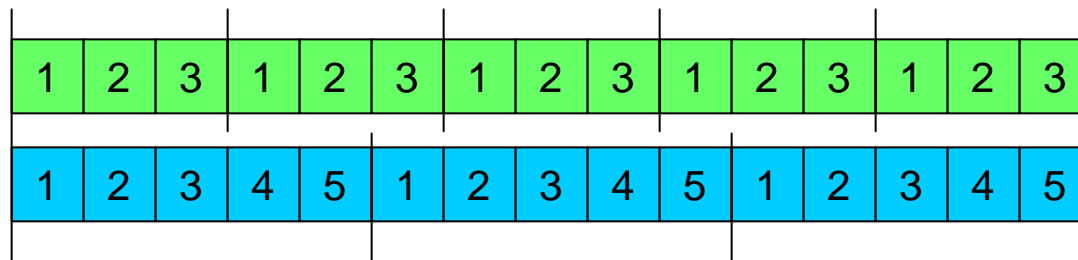


Combining Linear Congruential Generators

- Choose multiple generators with *relatively prime* cycle lengths:



- Overall sequence doesn't repeat until each individual generator has cycled a number of times, e.g.:



Tests for Random Numbers

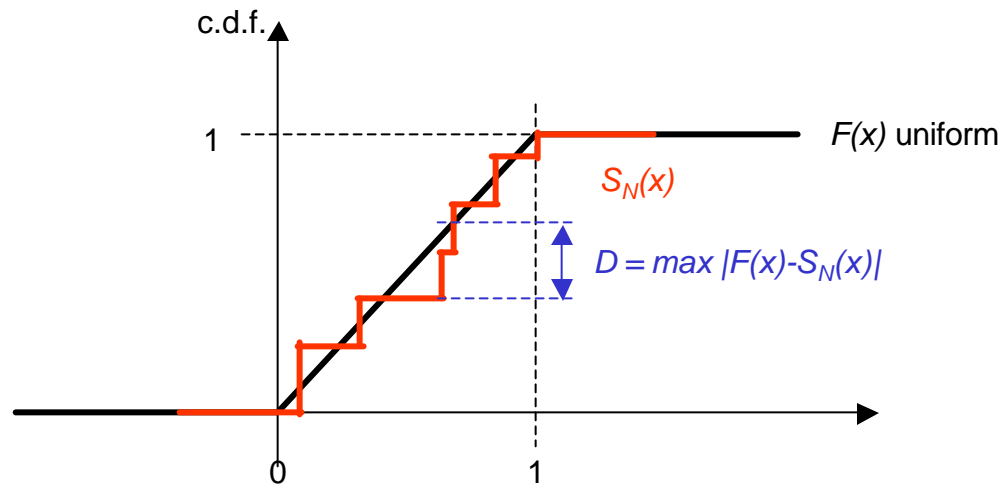
- Test for Uniformity:
 - Frequency test - Kolmogorov-Smirnov or χ^2 tests comparing the distribution to uniform
- Tests for Independence:
 - Runs test - looks for patterns of increasing/decreasing values
 - Autocorrelation test - calculate correlation between random numbers
 - Gap test - look for gaps between repetitions of a digit
 - Poker test - treats numbers as a poker hand and compare to expected hands.

Hypothesis Testing

- Uniformity:
 - H_0 , numbers are uniformly distributed
 - H_1 , numbers are not uniformly distributed
- Independence
 - H_0 , numbers are independently generated
 - H_1 , numbers are not independently generated
- Testing to reject the null hypothesis, but failure to reject H_0 does not prove H_1
- A level of significance, α , is specified for each test
$$\alpha = P(\text{reject } H_0 \mid H_0 \text{ true})$$
- Typically, $\alpha = .01, .05$ - which means that 1-5% of the test could indicate failure by chance.

Frequency Tests

- Kolmogorov-Smirnov test:
 - Compare empirical c.d.f. to expected c.d.f.



- Find maximum deviation, D
- For a given α , and N random values, find critical value from Table A.8
- For $N > 35$:

$D_{0.10}$	$D_{0.05}$	$D_{0.01}$
$\frac{1.22}{\sqrt{N}}$	$\frac{1.36}{\sqrt{N}}$	$\frac{1.63}{\sqrt{N}}$

Example 7.6

- Five random numbers, 0.44, 0.81, 0.14, 0.05, 0.93 are generated. Use K-S test with significance $\alpha=0.05$
- Order from smallest to largest, compare with expected c.d.f. of uniform distribution

$R_{(i)}$	0.05	0.14	0.44	0.81	0.93
i/N	0.20	0.40	0.60	0.80	1.00
$i/N - R_{(i)}$	0.15	0.26	0.16	-	0.07
$R_{(i)} - (i-1)/N$	0.05	-	0.04	0.21	0.13

critical value: $D_{0.05} = .565$

Distribution cannot be distinguished from uniform

Example 7.7

- Chi-square test

$$c^2 = \sum_{i=1}^n \frac{(O_i - E_i)^2}{E_i}$$

- O_i is the observed number in the i^{th} class
- E_i is the expected number in the i^{th} class
- n is number of classes
- pick the classes so each has at least 5 in class

- Use 10 uniform width classes $[0,.1)$, $[.1,.2)$, ... $[.9,1.0)$, so $E_i = 10$

- From data in example,
 $O_i = (8, 8, 10, 9, 12, 8, 10, 14, 10, 11)$
- $c^2 = 3.4$, which is less than critical value $c^2_{0.05,9} = 16.9$

Why use Frequency Tests?

- Consider the sequence of random numbers:

.03, .01, .04, .01,

.15, .19, .12, .16,

.25, .24, .23, .21,

.34, .31, .35, .39,

.42, .46, .45, .44,

...

.93, .91, .94, .91

- They would pass the K-S and chi-square test, but are not suitable as random numbers
- Frequency tests will find their deficiencies

Run Tests

- A *run* is a succession of similar events

- Coin flipping example:

HTTHHTTTHT

– six runs are marked

- For sequences of random numbers, define *up* runs and *down* runs, depending on whether successive numbers are increasing or decreasing. E.g.:

.87 .15 .23 .45 .69 .32 .30 .19 .24 .18 .65 .82 .93 .22 .81
↓ ↑ ↓ ↑ ↓ ↓ ↓ ↑ ↓ ↑ ↑ ↑ ↓ ↑

.08 .18 .23 .36 .42 .55 .63 .72 .89 .91
↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑

.08 .93 .15 .96 .26 .84 .28 .79 .36 .57
↑ ↓ ↑ ↓ ↑ ↓ ↑ ↓ ↑



These don't look like any realistic random sequence

Testing Runs

- For a truly random sequence with N samples, mean and variance of number of runs, a , are:

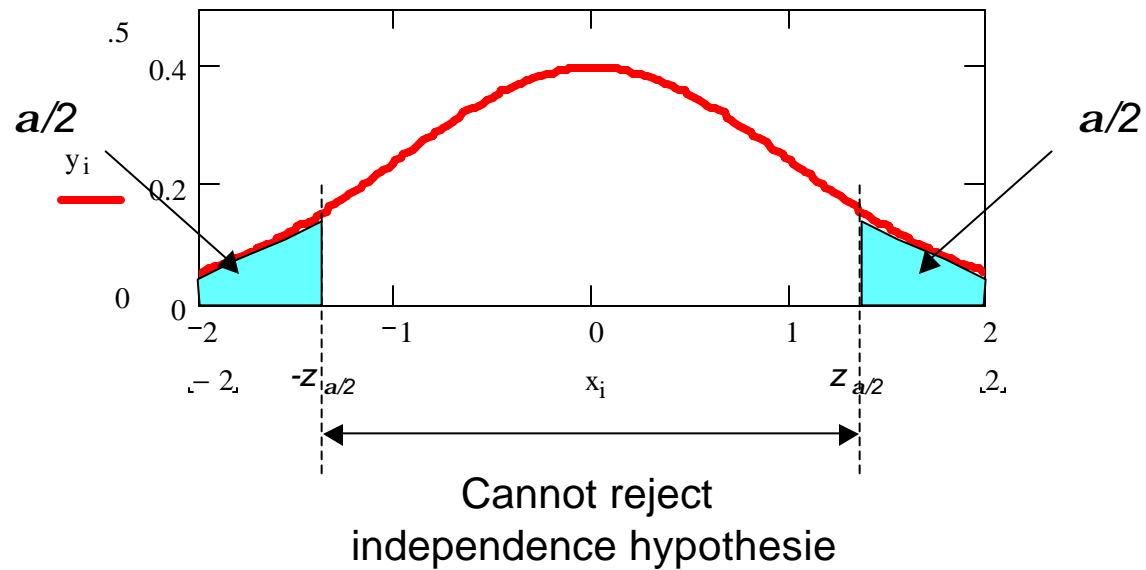
$$\mathbf{m}_a = \frac{2N-1}{3}$$
$$\mathbf{s}_a^2 = \frac{16N-29}{90}$$

- For $N > 20$, this can be approximated by a normal distribution, $N(\mathbf{m}_a, \mathbf{s}_a^2)$
- Observe the number of runs in the data, a
- Calculate test statistic:

$$Z_0 = \frac{a - \mathbf{m}_a}{\mathbf{s}_a} = \frac{a - [(2N-1)/3]}{\sqrt{(16N-29)/90}}$$

- Compare to a normal distribution

Testing Runs



- Find $z_{a/2}$ for significance $a/2$ from Normal distribution tables
- Check to see if statistic exceeds $z_{a/2}$

Homework

- Ch. 7 - exercises 7, 26 & 27