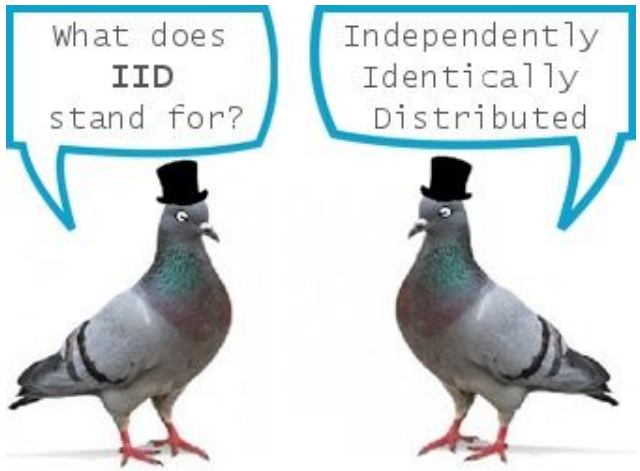
Random Numbers

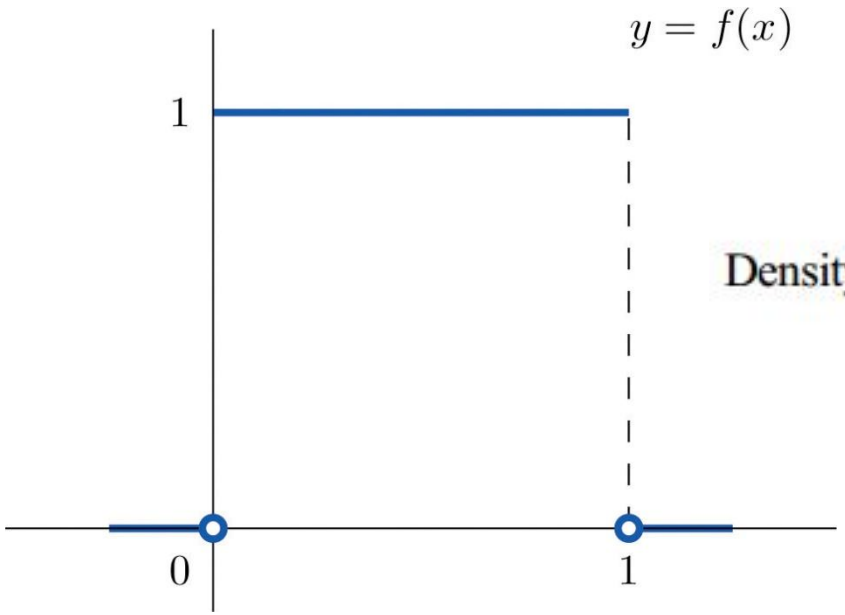
CSI 423: Simulation and Modelling

The goal

- Stochastic simulations need to generate IID (Independent and identically distributed) random numbers
  - Inter Arrival Times
  - Service times
- Specially we need U(0,1)



U(0,1)



Density function: $f(x) = \begin{cases} 1 & \text{if } 0 \le x \le 1 \\ 0 & \text{otherwise} \end{cases}$

Early Methods

- Physical
  - Cast lots
  - Dice
  - Cards
  - Urns
  - Lotteries
- Mechanical
  - Spinning disks
- Electrical
  - ERNIE
  - RAND
- Others
  - Pick digits "randomly" from Scottish phone directory or census reports
  - Decimals in expansion of π to 100,000 places

Algorithmic Methods

- Sequential
  - the next “random” number is determined by one or several of its predecessors according to a fixed mathematical formula
- Midsquare Method
  - von Neumann and Metropolis, 1945
  - Start with a 4 digit number
  - Square it and take middle 4 digits and put a decimal before the number

Midsquare Method

i	Z_i	U_i	Z_i^2
0	7182	—	51581124
1	5811	0.5811	33767721
2	7677	0.7677	58936329
3	9363	0.9363	87665769
4	6657	0.6657	44315649
5	3156	0.3156	09960336
6	9603	0.9603	92217609
7	2176	0.2176	04734976
8	7349	0.7349	54007801
9	0078	0.0078	00006084
10	0060	0.0060	00003600
11	0036	0.0036	00001296
12	0012	0.0012	00000144
13	0001	0.0001	00000001
14	0000	0.0000	00000000
15	0000	0.0000	00000000
.	.	.	.
.	.	.	.

Midsquare Method

- Disadvantages
  - Not really "random"—entire sequence determined by the seed Z
  - If a Z_i ever reappears, the entire sequence will be recycled
- We need long cycles

Can We Generate "Truly" Random Numbers?

- Only possible with physical experiment having output ~ U(0,1)
- Still some interest in this (counting gamma rays from space)
  - Not reproducible
  - Impractical for computers (wire in special circuits)
- **Practical view:** produce stream of numbers that appear to be IID

Criteria for RNG

1. Appear to be distributed uniformly on [0,1] and independent
2. Fast, low memory
3. Be able to reproduce- why?
    1. Makes debugging easier
    2. Comparison
4. Provision for separate streams

Linear Congruential Generators (LCG)

- Lehmer, 1954
- Specify four parameters
  - Z_0 = seed
  - m = modulus
  - a = multiplier
  - c = increment
- $Z_i=(aZ_{i-1}+c)(\text{mod } m)$
- $U_i = Z_i / m$

Linear Congruential Generators (LCG)

- Problems
  - Not really random
  - Cycles
  - They can take discrete values

Example

- m=16, a=5, c=3, z_0=7
- Cycle length = 16
- Its maximum – full period

i	Z_i	U_i
0	7	—
1	6	0.375
2	1	0.063
3	8	0.500
4	11	0.688
5	10	0.625
6	5	0.313
7	12	0.750
8	15	0.938
9	14	0.875
10	9	0.563
11	0	0.000
12	3	0.188
13	2	0.125
14	13	0.813
15	4	0.250
16	7	0.438
17	6	0.375
18	1	0.063
19	8	0.500

Full Period Theorem

- Hull and Dobell, 1966
- The **LCG $Z_i=(aZ_{i-1}+c)(\text{mod } m)$** has full period if and only if all three of the following hold
  1. c and m are relatively prime
  2. If q is any prime number that divides m, then q also divides a-1
  3. If 4 divides m, then 4 also divides a-1

Other properties

- Fast
- Low storage
- Reproducible
- Restart in the middle
- Multiple streams
- Portability Issue:
  - Might cause integer overflow

Some LCGs

Mixed ($c > 0$):

m	a	c
$2^{31} = 2{,}147{,}483{,}648$	314,159,269	453,806,245
$2^{35} = 34{,}359{,}738{,}368$	$5^{15} = 30{,}517{,}578{,}125$	1

Multiplicative ($c = 0$, the case for most LCGs):

m	a	
$2^{31} = 2{,}147{,}483{,}648$	$2^{16} + 3 = 65{,}539$	"RANDU," a ***terrible*** generator
$2^{31} - 1 = 2{,}147{,}483{,}647$	$7^5 = 16{,}807$	SIMAN, Arena, AweSim
	630,360,016	SIMSCRIPT, simlib
	742,938,285	GPSS/H
	397,204,094	GPSS/PC

General Congruences

- LCGs are special case of the form $Z_i=g(Z_{i-1},Z_{i-2},\ldots)$, $U_i=Z_i/m$

1. $g(Z_{i-1})=aZ_{i-1}+c$ (LCG)
2. $g(Z_{i-1}, Z_{i-2}, \ldots, Z_{i-q})= a_1Z_{i-1} + a_2Z_{i-2} + \ldots + a_qZ_{i-q}$ (multiple recursive generator)
3. $g(Z_{i-1})=a`Z_{i-1}^2+ aZ_{i-1}+ c$ (quadratic)
4. $g(Z_{i-1}, Z_{i-1})= Z_{i-1}+Z_{i-2}$ (fibonacci - bad)

Composite Generators

- Shuffling
  - Fill a vector of length 128 (say) from generator 1
  - Use generator 2 to pick one of the 128 in the vector
  - Fill the hole with the next value from generator 1, use generator 2 to pick one of the 128 in the vector, etc.
  - shuffling a bad generator improves it, but shuffling a good generator doesn't gain much.

Composite Generators

- Differencing LCGs
  - Let Z_{1i} and Z_{2i} from different LCGs with different moduli
  - Let $Z_i=(Z_{1i}-Z_{2i})(\text{mod } m)$; $U_i=Z_i/m$
  - Very long period
  - Good statistical properties
  - portable

Composite Generators

- Wichman/Hill
  - Use three LCGs to get U_{1i},U_{2i} and U_{3i}
  - Let U_i= sum of fractional part of U_{1i},U_{2i} and U_{3i}
  - Good period
  - But equivalent to LCGs

Combined Multiple Recursive Generators

Recall the single MRG: $Z_i = (a_1 Z_{i-1} + a_2 Z_{i-2} + \ldots + a_q Z_{i-q}) \pmod{m}$, $U_i = Z_i/m$

Have J MRGs running simultaneously: $\{Z_{1,i}\}, \{Z_{2,i}\}, \ldots, \{Z_{J,i}\}$

Let m_1 be the modulus used in the first of these J MRGs

For constants $\delta_1, \delta_2, \ldots, \delta_J$, define

$$Y_i = (\delta_1 Z_{1,i} + \delta_2 Z_{2,i} + \ldots + \delta_J Z_{J,i}) \pmod{m_1}$$

Return $U_i = Y_i / m$

Must choose constants carefully, but extremely long periods and extremely good statistical behavior can be achieved

Period is approximately 3.1×10^{57}

Tausworthe

- Originated in cryptography
- Generate sequence of bits b_1, b_2, b_3, ... via congruence

$$b_i = (b_{i-r} + b_{i-q}) \pmod 2 = \begin{cases} 0 \text{ if } b_{i-r} = b_{i-q} \\ 1 \text{ if } b_{i-r} \neq b_{i-q} \end{cases}$$

- Various algorithms to group bits into U_i's
- Can achieve very long periods

Unpredictable Generators

- Blum/Blum/Shub
- Another way to generate a sequence of bits
- Pick p and q to be large (like 40-digit) prime numbers, set m = pq
- $X_i = X_{i-1}^2 \pmod{m}$
- b_i = parity of X_i (0 if even, 1 if odd)
- P=11, q=19 seed=3

Testing Random Numbers

- Chi Square test

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

Chi Square (χ^2) Test

- We test if the observed data fits any distribution using Chi Square Statistic
- Carl Pearson, 1900

Example

- Suppose a restaurant manager is making the following claim about number of customers
- We have to test whether he is saying truth or not

Day	Sat	Sun	Mon	Tue	Wed	Thu
# of Customers	10%	10%	15%	20%	30%	15%

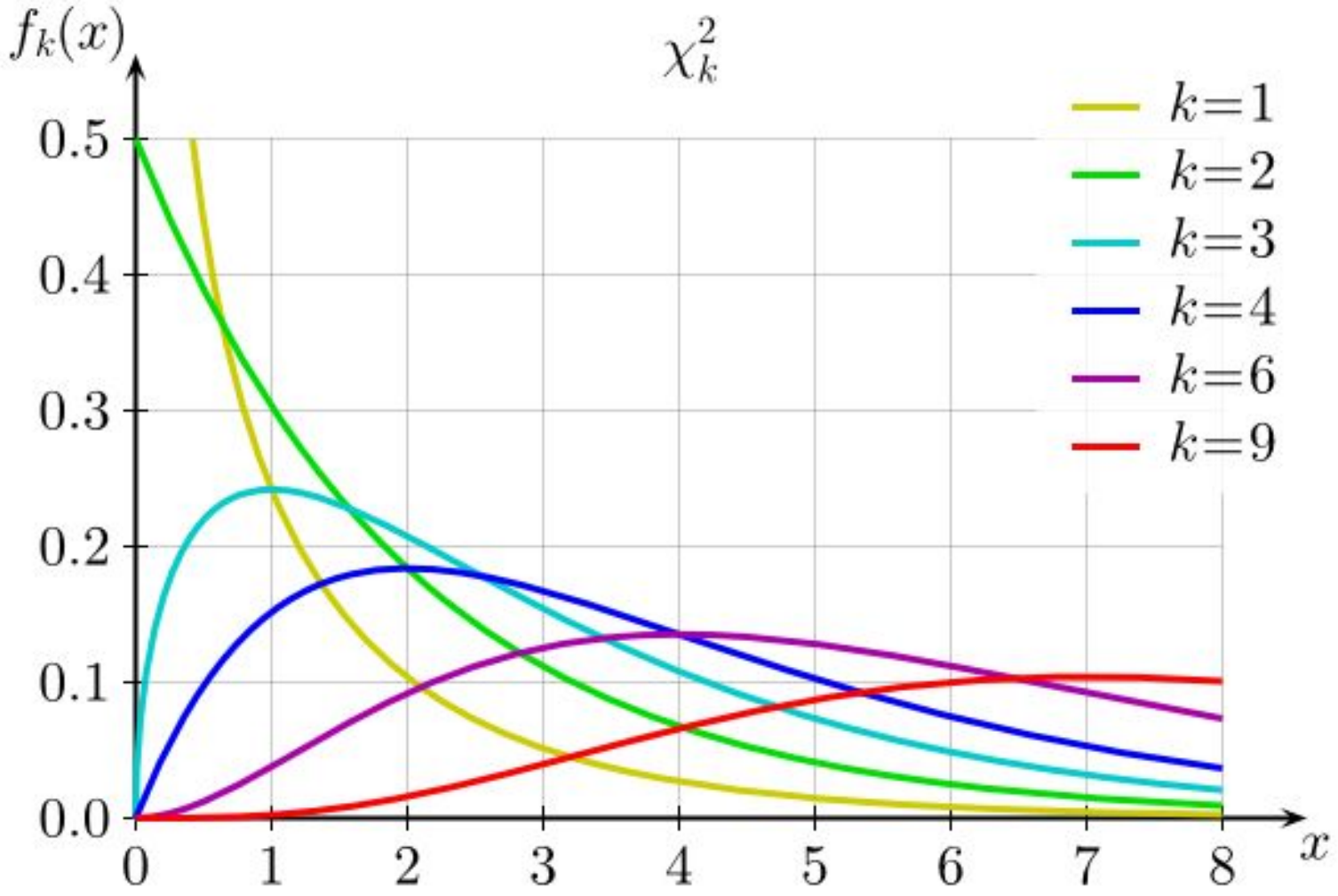
Null Hypothesis

- H_0= Owner's hypothesis is correct
- Alternate hypothesis: H_1 = Owner's hypothesis is not correct
- Suppose we observe the following data:

Day	Sat	Sun	Mon	Tue	Wed	Thu	Total
# of Customers	30	14	34	45	57	20	200
Expected	20	20	30	40	60	30	

X^2 calculation

- $X^2 = (30-20)^2/20 + (14-20)^2/20 + (34-30)^2/30 + (45-40)^2/40 + (57-60)^2/60 + (40-30)^2/30 = 11.44$
- Now we have to test what is the probability to get this value?
- Degrees of freedom = 6-1=5


$f_k(x)$
χ^2_k
$k=1$
$k=2$
$k=3$
$k=4$
$k=6$
$k=9$
0.5
0.4
0.3
0.2
0.1
0.0
0
1
2
3
4
5
6
7
8
x

Percentage Points of the Chi-Square Distribution

Degrees of Freedom	Probability of a larger value of x^2								
	0.99	0.95	0.90	0.75	0.50	0.25	0.10	0.05	0.01
1	0.000	0.004	0.016	0.102	0.455	1.32	2.71	3.84	6.63
2	0.020	0.103	0.211	0.575	1.386	2.77	4.61	5.99	9.21
3	0.115	0.352	0.584	1.212	2.366	4.11	6.25	7.81	11.34
4	0.297	0.711	1.064	1.923	3.357	5.39	7.78	9.49	13.28
5	0.554	1.145	1.610	2.675	4.351	6.63	9.24	11.07	15.09
6	0.872	1.635	2.204	3.455	5.348	7.84	10.64	12.59	16.81
7	1.239	2.167	2.833	4.255	6.346	9.04	12.02	14.07	18.48
8	1.647	2.733	3.490	5.071	7.344	10.22	13.36	15.51	20.09
9	2.088	3.325	4.168	5.899	8.343	11.39	14.68	16.92	21.67
10	2.558	3.940	4.865	6.737	9.342	12.55	15.99	18.31	23.21
11	3.053	4.575	5.578	7.584	10.341	13.70	17.28	19.68	24.72
12	3.571	5.226	6.304	8.438	11.340	14.85	18.55	21.03	26.22
13	4.107	5.892	7.042	9.299	12.340	15.98	19.81	22.36	27.69
14	4.660	6.571	7.790	10.165	13.339	17.12	21.06	23.68	29.14
15	5.229	7.261	8.547	11.037	14.339	18.25	22.31	25.00	30.58
16	5.812	7.962	9.312	11.912	15.338	19.37	23.54	26.30	32.00
17	6.408	8.672	10.085	12.792	16.338	20.49	24.77	27.59	33.41
18	7.015	9.390	10.865	13.675	17.338	21.60	25.99	28.87	34.80
19	7.633	10.117	11.651	14.562	18.338	22.72	27.20	30.14	36.19
20	8.260	10.851	12.443	15.452	19.337	23.83	28.41	31.41	37.57
22	9.542	12.338	14.041	17.240	21.337	26.04	30.81	33.92	40.29
24	10.856	13.848	15.659	19.037	23.337	28.24	33.20	36.42	42.98
26	12.198	15.379	17.292	20.843	25.336	30.43	35.56	38.89	45.64
28	13.565	16.928	18.939	22.657	27.336	32.62	37.92	41.34	48.28
30	14.953	18.493	20.599	24.478	29.336	34.80	40.26	43.77	50.89
40	22.164	26.509	29.051	33.660	39.335	45.62	51.80	55.76	63.69
50	27.707	34.764	37.689	42.942	49.335	56.33	63.17	67.50	76.15
60	37.485	43.188	46.459	52.294	59.335	66.98	74.40	79.08	88.38

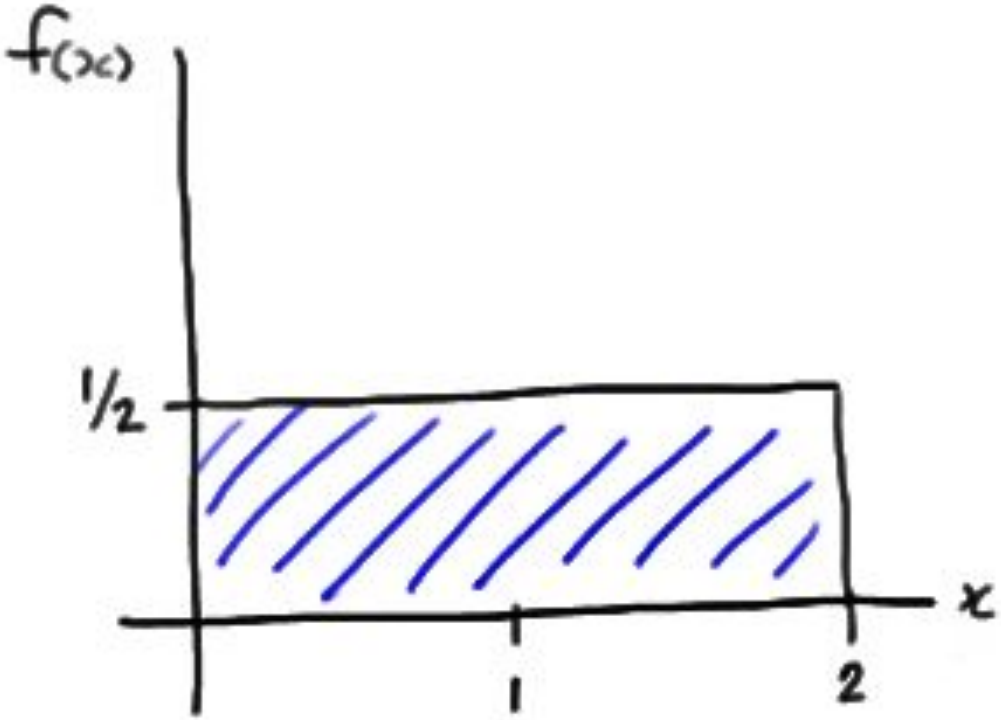
Kolmogorov Smirnov Test

- KS tests the hypothesis that a given unknown distribution Ψ is equal to another distribution, Θ
- Null Hypothesis, $H_0: \Psi = \Theta$
- Alternate hypothesis, $H_1: \Psi \neq \Theta$

KS Test Example

- Suppose we observe 8 data points
  - **1.41 0.26 1.97 0.33**
    0.55 0.77 1.46 1.18
- Is there any evidence to suggest that the data were not randomly sampled from a Uniform(0, 2) distribution?

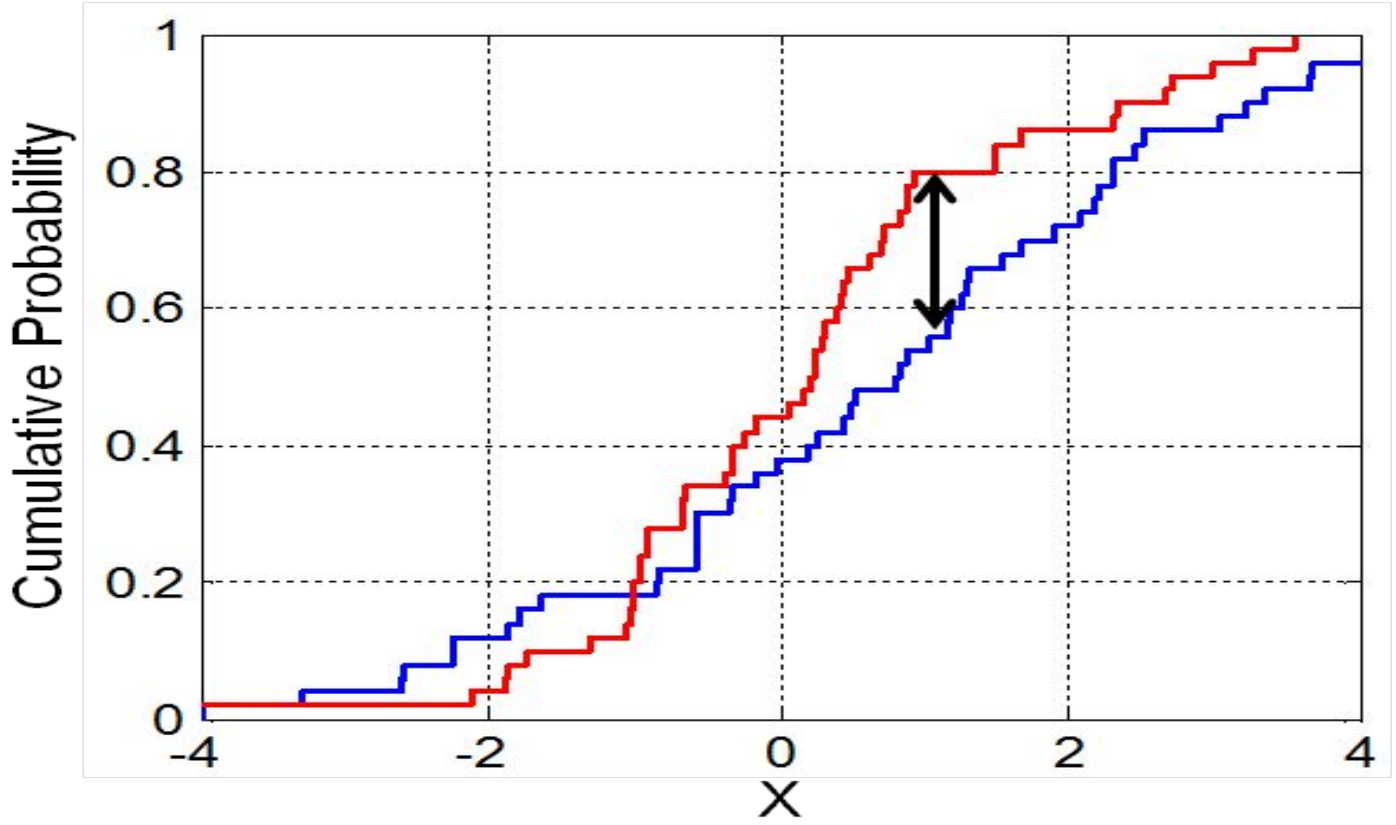
U(0,2)



U(0,2)

$$F_0(x) = \begin{cases} 0, & \text{for } x < 0 \\ \frac{1}{2}x, & \text{for } 0 \leq x < 2 \\ 1, & \text{for } x \geq 2 \end{cases}$$

KS Test



KS Test

k	y_k	$F_n(y_{k-1})$	$F_0(y_k)$	$F_n(y_k)$	$\lvert F_n(y_{k-1}) - F_0(y_k)\rvert$	$\lvert F_0(y_k) - F_n(y_k)\rvert$
1	0.26	0.000	0.130	0.125	0.130	0.005
2	0.33	0.125	0.165	0.250	0.040	0.085
3	0.55	0.250	0.275	0.375	0.025	0.100
4	0.77	0.375	0.385	0.500	0.010	0.115
5	1.18	0.500	0.590	0.625	0.090	0.035
6	1.41	0.625	0.705	0.750	0.080	0.045
7	1.46	0.750	0.730	0.875	0.020	**0.145**
8	1.97	0.875	0.985	1.000	0.090	0.015

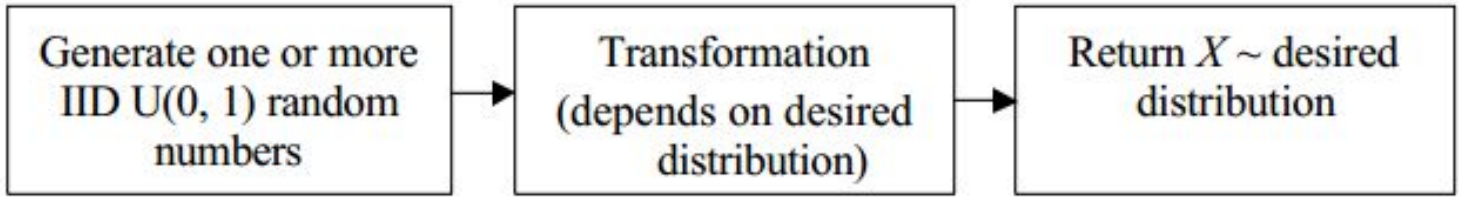
$$D_n = \sup_x[|F_n(x) - F_0(x)|]$$

$$\alpha = 1 - P(D_n \leq d)$$

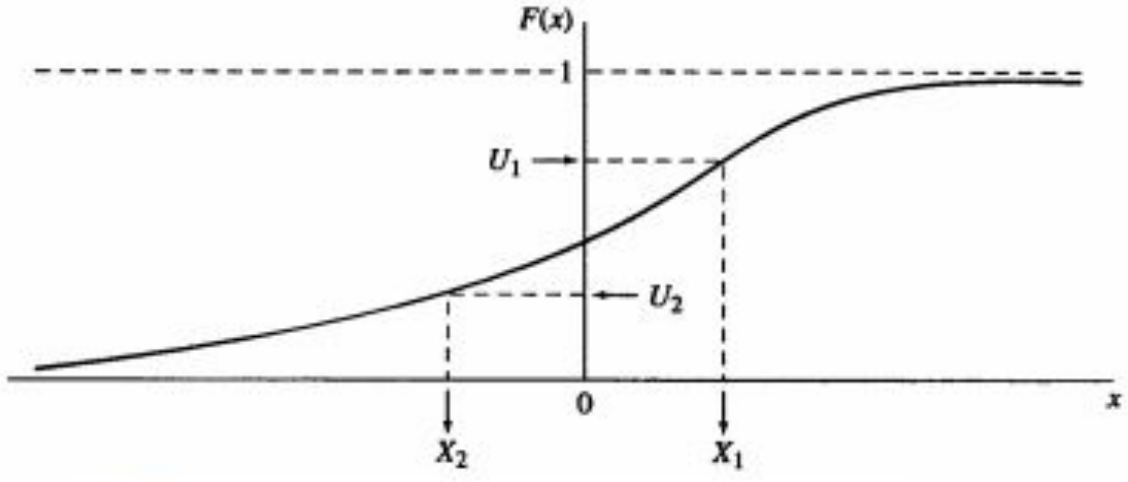
	α			
n	0.20	0.10	0.05	0.01
1	0.90	0.95	0.98	0.99
2	0.68	0.78	0.84	0.93
3	0.56	0.64	0.71	0.83
4	0.49	0.56	0.62	0.73
5	0.45	0.51	0.56	0.67
6	0.41	0.47	0.52	0.62
7	0.38	0.44	0.49	0.58
8	0.36	0.41	0.46	0.54
9	0.34	0.39	0.43	0.51
10	0.32	0.37	0.41	0.49

Variates

- observations ("variates") from some desired input distribution (exponential, gamma, etc.)
- Inverse Transform
- Composition
- Convolution



Inverse Transform (continuous)



Algorithm:

1. Generate $U \sim \mathrm{U}(0, 1)$ (random-number generator)
2. Find X such that $F(X) = U$ and return this value X

Weibull Distribution

Density function is $f(x) = \begin{cases} \alpha\beta^{-\alpha}x^{\alpha-1}e^{-(x/\beta)^{\alpha}} & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$

CDF is $F(x) = \int_{-\infty}^{x} f(t)dt = \begin{cases} 1 - e^{-(x/\beta)^{\alpha}} & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$

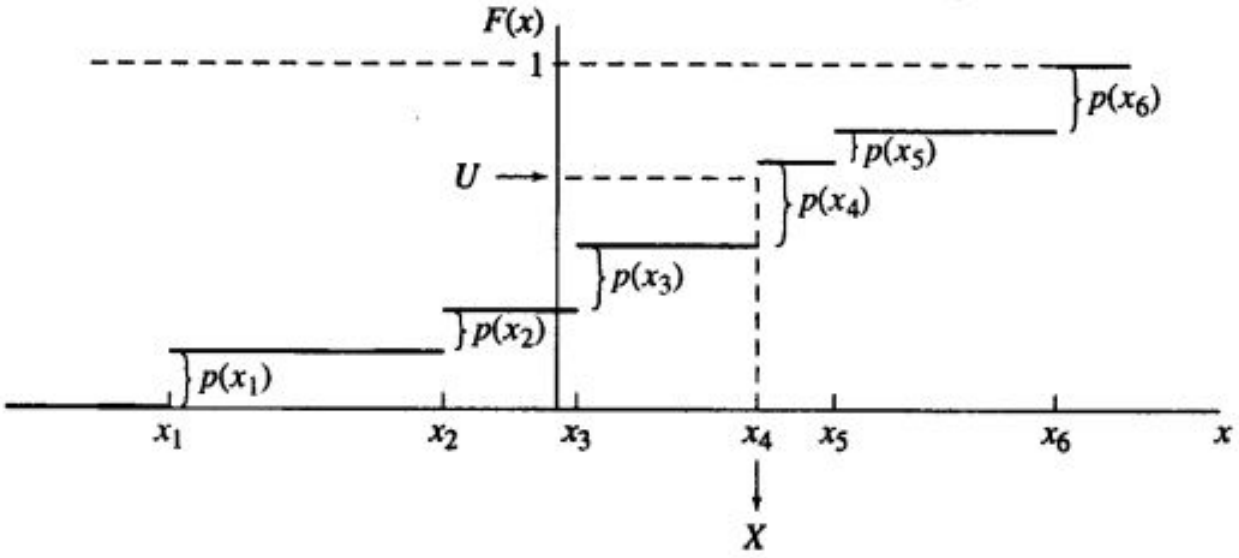
Solve $U = F(X)$ for X:

$$\begin{aligned} U &= 1 - e^{-(X/\beta)^{\alpha}} \\ e^{-(X/\beta)^{\alpha}} &= 1 - U \\ -(X/\beta)^{\alpha} &= \ln(1-U) \\ X/\beta &= [-\ln(1-U)]^{1/\alpha} \\ X &= \beta[-\ln(1-U)]^{1/\alpha} \end{aligned}$$

Since $1 - U \sim \mathrm{U}(0, 1)$ as well, can replace $1 - U$ by U to get the final algorithm:

1. Generate $U \sim \mathrm{U}(0, 1)$
2. Return $X = \beta(-\ln U)^{1/\alpha}$

Inverse Transform (Discrete)



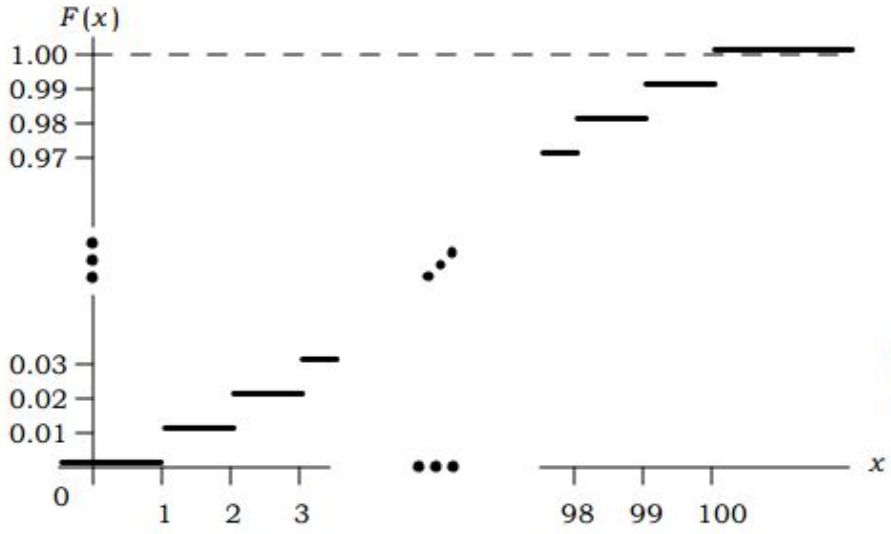
Algorithm:

1. Generate $U \sim \mathrm{U}(0,1)$ (random-number generator)
2. Find the smallest positive integer I such that $U \le F(x_I)$
3. Return $X = x_I$

Inverse Transform (Discrete)

Discrete uniform distribution on 1, 2, ..., 100

$x_i = i$, and $p(x_i) = p(i) = P(X = i) = 0.01$ for $i = 1, 2, ..., 100$



1. Generate $U \sim U(0,1)$
2. If $U \le 0.01$ return $X = 1$ and stop; else go on
3. If $U \le 0.02$ return $X = 2$ and stop; else go on
4. If $U \le 0.03$ return $X = 3$ and stop; else go on

.
.
.

100. If $U \le 0.99$ return $X = 99$ and stop; else go on
101. Return $X = 100$

Inverse Transform Method

- Limitations
  - Must invert CDF, which may be difficult (numerical methods)
  - May not be the fastest or simplest approach for a given distribution
- Advantages
  - Uses only a single U(0,1)
- Example / Try Others
  - Bernoulli, Poisson, Exponential

Composition

Suppose we can find other CDFs F_1, F_2, ... (finite or infinite list) and weights p_1, p_2, ... ($p_j \geq 0$ and $p_1 + p_2 + \cdots = 1$) such that for all x,

$$F(x) = p_1 F_1(x) + p_2 F_2(x) + \cdots$$

(Equivalently, can decompose density $f(x)$ or mass function $p(x)$ into convex combination of other density or mass functions)

Algorithm:

1. Generate a positive random integer J such that $P(J = j) = p_j$
2. Return X with CDF F_J (given $J = j$, X is generated independent of J)

HyperExponential Distribution

Let $X \sim$ Hyperexponential$(\lambda_1, \alpha_1, \lambda_2, \alpha_2)$ so that

$$f_x(x) = \alpha_1 \lambda_1 e^{-\lambda_1 x} + \alpha_2 \lambda_2 e^{-\lambda_2 x}.$$

In our earlier notation we have

$$\begin{aligned} \alpha_1 &= p_1 \\ \alpha_2 &= p_2 \\ f_1(x) &= \lambda_1 e^{-\lambda_1 x} \\ f_2(x) &= \lambda_2 e^{-\lambda_2 x} \end{aligned}$$

generate U_1
if $U_1 \leq p_1$ **then**
    set $i = 1$
else
    set $i = 2$
generate U_2
/* Now generate X from Exp(λ_i) */
set

$$X = -\frac{1}{\lambda_i} \log(U_2)$$

Convolution

Suppose desired RV X has same distribution as $Y_1 + Y_2 + \cdots + Y_m$, where the Y_j's are IID and m is fixed and finite

Write: $X \sim Y_1 + Y_2 + \cdots + Y_m$, called *m-fold convolution* of the distribution of Y_j

Contrast with composition:

Composition: Expressed the *distribution function* (or density or mass) as a (weighted) sum of other distribution functions (or densities or masses)

Convolution: Express the *random variable itself* as the sum of other random variables

Algorithm (obvious):

1. Generate $Y_1, Y_2, \cdots, Y_m$ independently from their distribution
2. Return $X = Y_1 + Y_2 + \cdots + Y_m$

Acceptance-Rejection

Let X be a random variable with density, $f(\cdot)$, and CDF, $F_x(\cdot)$. Suppose it's hard to simulate a value of X directly using either the inverse transform or composition algorithm. We might then wish to use the acceptance-rejection algorithm.

Let Y be another random variable with density $g(\cdot)$ and suppose that it is easy to simulate a value of Y. If there exists a constant a such that

$$\frac{f(x)}{g(x)} \leq a \text{ for all } x$$

then we can simulate a value of X as follows.

The Acceptance-Rejection Algorithm

generate Y with PDF $g(\cdot)$
generate U
while $U > \frac{f(Y)}{ag(Y)}$
    generate Y
    generate U
set $X = Y$

Reading

- Chapter 7,8
  - Law Kelton Book