

Chapter 2

Simulation Examples

Goals of the Chapter

- This chapter presents several examples of simulations that can be performed by **devising a simulation table** either manually or *with a spreadsheet*.
- The simulation table provides a systematic method for tracking system state over time.
- These examples provide insight into the methodology of discrete-system simulation and the *descriptive statistics* used for predicting system performance.
- This chapter gives a number of simulation examples in queueing, inventory, reliability, and network analysis.

Spreadsheet Examples

- Example 2.1: Coin Toss
- Example 2.2: Random Service Times
- Example 2.3: Random Arrival Times
- Cumulative Distribution Function (cdf) is denoted by $F(x)$, measures the probability that the random variable $X \leq x$, i.e., $F(x) = P(X \leq x)$
 - If X is discrete, then:
 - Let $p(x_i)$ = probability the random variable is $x_i = P(X = x_i)$

$$F(x) = \sum_{\substack{\text{all} \\ x_i \leq x}} p(x_i)$$

Spreadsheet Examples (Example 2.1)



- Example 2.1: Coin Toss
 - Simulates a coin-tossing experiment
 - Solution uses a uniform random number generator

Spreadsheet Examples (Example 2.2)

■ Example 2.2: Random Service Times

- Simulates call-processing times of an automated telephone information system – **activity times**
- Assumes information server spends 3, 6 or 10 minutes with each caller
- Assumes proportion of calls to each type of 30%, 45% and 25%
- Generates “random samples” from a given distribution (empirical)

Spreadsheet Examples (Example 2.3)

■ Example 2.3: Random Arrival Times

- Generate random inter-arrival times (i.e., time between successive arrivals) – **activity times**
 - Actual arrival times (time of arrival) is determined by the “simulation clock”
- Simulates telephone calls with inter-arrival times of 1, 2, 3 or 4 minutes
- Discrete Uniform distribution [1,4] minutes, all values are equally likely.

Simulation Table: Steps to follow

1. Determine the characteristics of each of the inputs to the simulation. Quite often, these are modeled as probability distributions, either continuous or discrete.
2. Construct a simulation table. Each simulation table is different, for each is developed for the problem at hand.
3. Each column in the table is one of the following:
 - ☐ An activity time associated with model input
 - ☐ Any other random variable defined by a model input
 - ☐ A system state
 - ☐ An event, or the clock tick of an event
 - ☐ A model output
 - ☐ A model response

Steps to follow (continued ...)

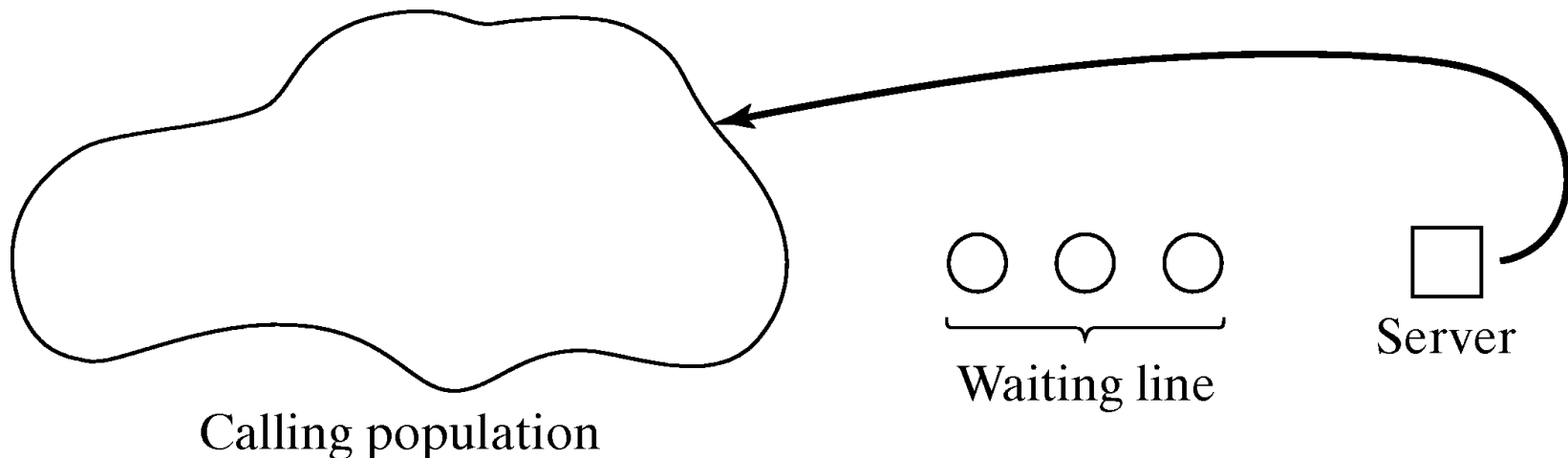
A Generic simulation Table:

- Let there be p inputs, $x_{ij}, j = 1, 2, \dots, p$, and one response, Y_i , for each of repetitions (or, trials) $i = 1, 2, \dots, n$.
- Initialize the table by filling in the data for repetition 1
- For each repetition i , generate a value for each of the p inputs, and evaluate the function, calculating a value of the response Y_i .
- The input values may be computed by sampling values from the distributions chosen in step 1.
- A response typically depends on the inputs and one or more previous responses.

Step	Activities and system states	Output
	$x_{i1} \ x_{i2} \ x_{i3} \ \dots \ x_{ij} \ \dots \ x_{ip}$	y_i
1		
n		

Simulation of Queueing Systems

- A queueing system is described :
 - by its calling population, the nature of arrivals, the service mechanism, the system capacity and the queue and any priorities within the queue or service



Single Channel (Server) Queue

- In the single-channel queue, *the calling population is infinite*; that is, if a unit leaves the calling population and joins the waiting line or enters service, *there is no change in the arrival rate* of other units that could need service.
- *Arrivals for service occur one at a time* in a random fashion; once they join the waiting line, they are eventually served.
- In addition, *service times are of some random length* according to a probability distribution which does not change over time.
- *The system capacity has no limit*, meaning that any number of units can wait in line

Single Channel (Server) Queue (contd ..)

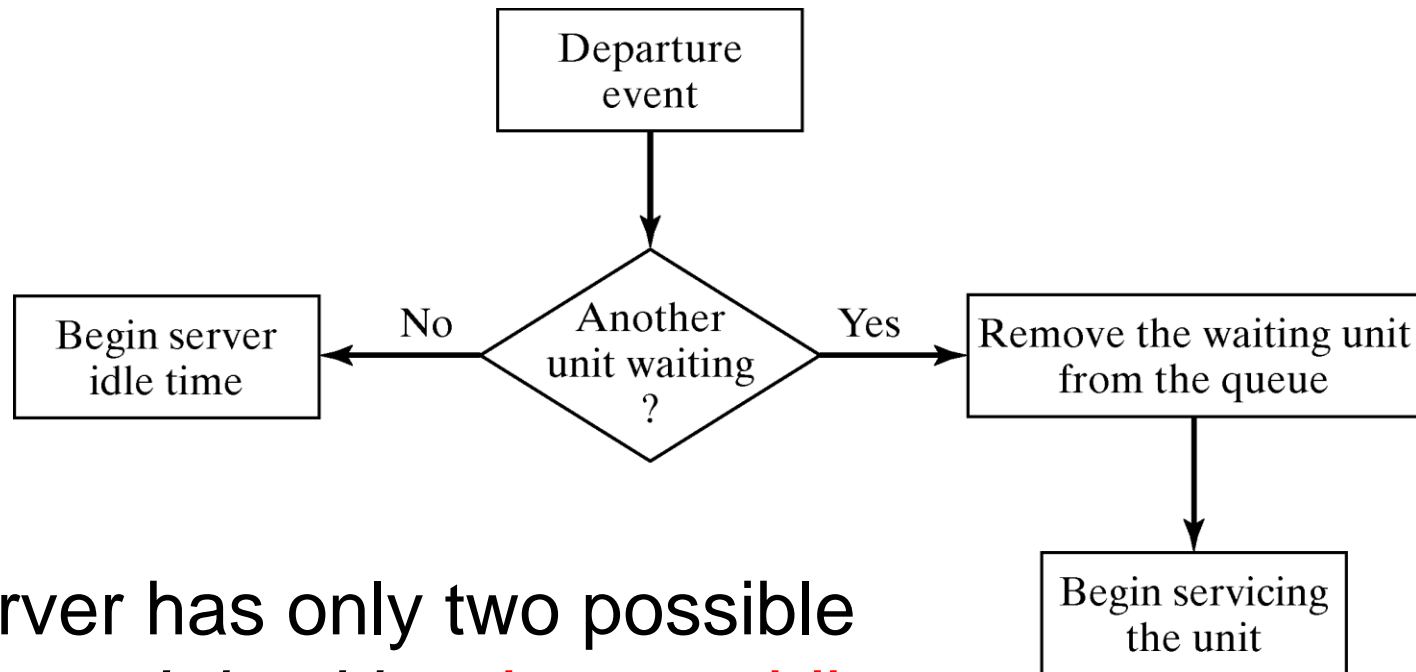
- Finally, **units are served in the order of their arrival** (often called FIFO: first in, first out) by a single server or channel.
- Arrivals and services are defined by the distribution of the time between arrivals and the distribution of service times, respectively.
- For any simple single or multi-channel queue, ***the overall effective arrival rate must be less than the total service rate***, or the waiting line will grow without bound.
- When queues grow without bound, they are termed "**explosive**" or unstable.

Single Server Queue (continued ..)

- Prior to our introducing several simulations of queueing systems, it is necessary to understand the concepts of *system state, events, and simulation clock*.
- The *state of the system* is the *number of units* in the system and the *status* of the server, busy or idle.
- An *event* is a set of circumstances that causes an instantaneous change in the state of the system.
- In a single-channel queueing system, there are only two possible events that can affect the state of the system.
 - *The entry* of a unit into the system (the arrival event)
 - *The completion* of service on a unit (the departure event).
- The queueing system includes the server, the unit being serviced (if one is being serviced), and the units in the queue (if any are waiting).
- A *simulation clock* is used to track *simulated time*

Single Channel Queue – Departure Event

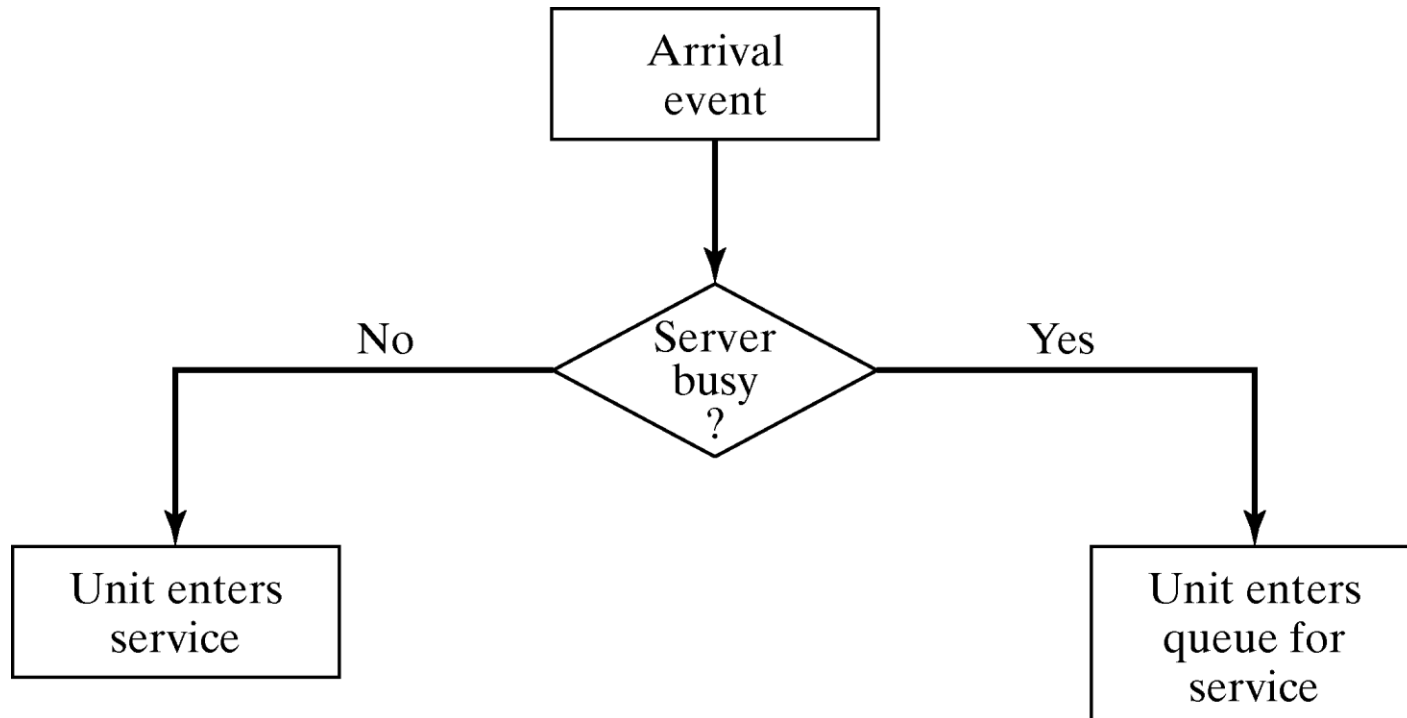
- If a unit has just completed service, the simulation proceeds in the manner shown in this flow diagram



Server has only two possible states: *It is either **busy** or **idle**.*

Single Channel Queue – Arrival Event

- The arrival event occurs when a unit enters the system. The flow diagram for the arrival event is shown below.


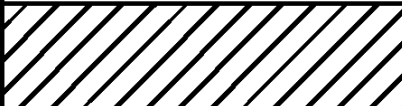


Unit Status

Arrival Event

		Queue status	
		Not empty	Empty
Server status	Busy	Enter queue	Enter queue
	Idle	Impossible	Enter service

Departure Event

		Queue status	
		Not empty	Empty
Server outcomes	Busy		Impossible
	Idle	Impossible	

Simulated Clock

- How can the events described above occur in simulated time?
- Simulations of queueing systems generally require the maintenance of an *event list* for determining what happens next.
- The *event list* tracks the *future times* at which the different types of events occur.
- Simulations using event lists are described in Chapter 3.
- Here the simulation is simplified by tracking each unit explicitly.
- In simulation, events usually occur at random times, the randomness imitating uncertainty in real life
- The randomness needed to imitate real life is made possible through the use of “Random Numbers”
- Random numbers are distributed uniformly and independently on the interval (0, 1)

Random Numbers

- Random numbers also can be generated in simulation packages and in spreadsheets (such as Excel).
- For example, Excel has a macro function called RAND() that returns a "random" number between 0 and 1.
- When numbers are generated by using a procedure, they are often referred to as *Pseudo-Random Numbers (PRNs)*
 - Because the procedure is fully known, it is always possible to predict the sequence of numbers that will be generated prior to the simulation
- We will discuss random number generation later

Single Channel Queue – Interarrival Times

- In a single-channel queueing simulation, interarrival times and service times are generated from the distributions of these random variables.
- For simplicity, assume that the times between arrivals were generated by **rolling a dice (die) five times** and recording the up face. Table below contains a set of five inter-arrival times generated in this manner.

<i>Customer</i>	<i>Interarrival Time</i>	<i>Arrival Time on Clock</i>
1	—	0
2	2	2
3	4	6
4	1	7
5	2	9
6	6	15

Interarrival Times (continued ..)

- These five interarrival times are used to compute the arrival times of six customers at the queueing system.
- The first customer is assumed to arrive at clock time 0. This starts the clock in operation.
- The second customer arrives two time units later, at clock time 2.
- The third customer arrives four time units later, at clock time 6; and so on

Single Channel Queue – Service Times

- The second time of interest is the service time.
- Table below contains service times generated at random from a distribution of service times.
- The only possible service times are one, two, three, and four time units, all four values are equally likely to occur

These random numbers are generated by placing the numbers one through four on chips and drawing the chips from a hat with replacement, being sure to record the numbers selected

<i>Customer</i>	<i>Service Time</i>
1	2
2	1
3	3
4	2
5	1
6	4

Single Channel Queue – Simulation Table

- Now, the interarrival times and service times must be meshed (stitched) to simulate the single-channel queueing system
- Table 2.7 was designed specifically for a single-channel queue that serves customers on a first-in-first-out (FIFO) basis

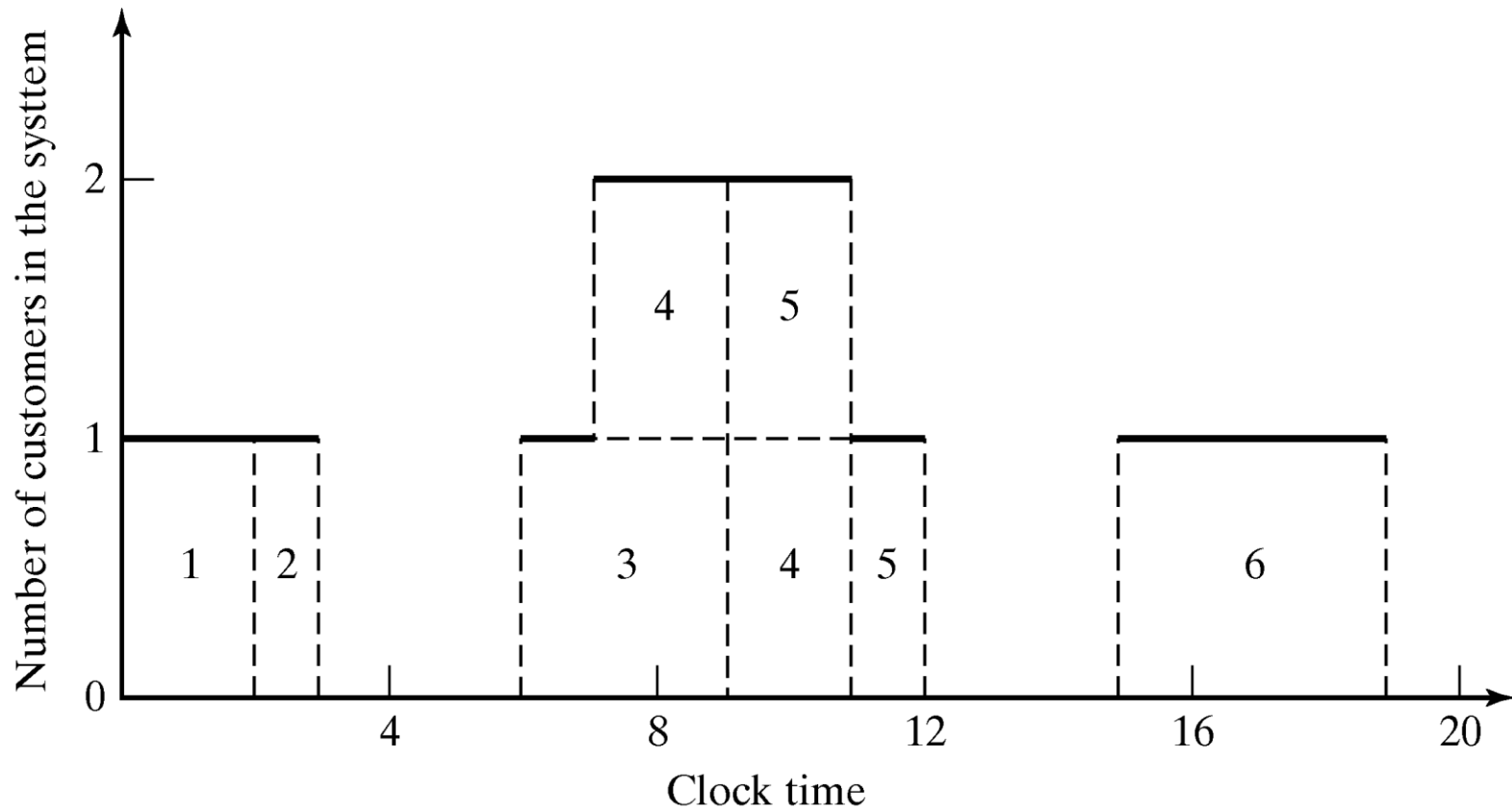
A	B	C	D	E
<i>Customer Number</i>	<i>Arrival Time (Clock)</i>	<i>Time Service Begins (Clock)</i>	<i>Service Time (Duration)</i>	<i>Time Service Ends (Clock)</i>
1	0	0	2	2
2	2	2	1	3
3	6	6	3	9
4	7	9	2	11
5	9	11	1	12
6	15	15	4	19

Chronological Ordering of Events

- Table 2.8 is ordered by clock time, in which case the events may or may not be ordered by customer number. The **chronological ordering of events (event list) is the basis** of the approach to discrete-event simulation

<i>Event Type</i>	<i>Customer Number</i>	<i>Clock Time</i>
Arrival	1	0
Departure	1	2
Arrival	2	2
Departure	2	3
Arrival	3	6
Arrival	4	7
Departure	3	9
Arrival	5	9
Departure	4	11
Departure	5	12
Arrival	6	15
Departure	6	19

Number of Customers in the System



Grocery Store Example (Example 2.5)

- A small grocery store has only one checkout counter.
- Customers arrive at this checkout counter at random times that are from 1 to 8 minutes apart, each possible value of interarrival time has the same probability of occurrence, as shown in Table 2.9
- The service times vary from 1 to 6 minutes, with the probabilities shown in Table 2.10
- Simulate up to 100 customers (too small a data for good analysis)

Table 2.6 Distribution of Time Between Arrivals

<i>Time between Arrivals (Minutes)</i>	<i>Probability</i>	<i>Cumulative Probability</i>	<i>Random Digit Assignment</i>
1	0.125	0.125	001–125
2	0.125	0.250	126–250
3	0.125	0.375	251–375
4	0.125	0.500	376–500
5	0.125	0.625	501–625
6	0.125	0.750	626–750
7	0.125	0.875	751–875
8	0.125	1.000	876–000

Average Interval Arrival Time = $(1+8)/2 = 4.5$ minutes

Table 2.7 Service-Time Distribution

<i>Service Time (Minutes)</i>	<i>Probability</i>	<i>Cumulative Probability</i>	<i>Random Digit Assignment</i>
1	0.10	0.10	01–10
2	0.20	0.30	11–30
3	0.30	0.60	31–60
4	0.25	0.85	61–85
5	0.10	0.95	86–95
6	0.05	1.00	96–00

Average Service time = $\sum s.p(s) = 3.2$ minutes

Interarrival and Service Time Generation

Table 2.8 Time-Between-Arrival Determination

<i>Customer</i>	<i>Random Digits</i>	<i>Time between Arrivals (Minutes)</i>	<i>Customer</i>	<i>Random Digits</i>	<i>Time between Arrivals (Minutes)</i>
1	—	—	11	413	4
2	064	1	12	462	4
3	112	1	13	843	7
4	678	6	14	738	6
5	289	3	15	359	3
6	871	7	16	888	8
7	583	5	17	902	8
8	139	2	18	212	2
9	423	4	⋮	⋮	⋮
10	039	1	100	538	5

Table 2.9 Service Times Generated

<i>Customer</i>	<i>Random Digits</i>	<i>Service Time (Minutes)</i>	<i>Customer</i>	<i>Random Digits</i>	<i>Service Time (Minutes)</i>
1	84	4	11	94	5
2	18	2	12	32	3
3	87	5	13	79	4
4	81	4	14	92	5
5	06	1	15	46	3
6	91	5	16	21	2
7	79	4	17	73	4
8	09	1	18	55	3
9	64	4	⋮	⋮	⋮
10	38	3	100	26	2

Grocery Store Simulation Table

Table 2.10 Simulation Table for Single-Channel Queuing Problem

<i>Customer</i>	<i>Interarrival Time (Minutes)</i>	<i>Clock Arrival Time</i>	<i>Service Time (Minutes)</i>	<i>Clock Time Service Begins</i>	<i>Waiting Time in Queue (Minutes)</i>	<i>Clock Time Service Ends</i>	<i>Time Customer Spends in System (Minutes)</i>	<i>Idle Time of Server (Minutes)</i>
1		0	4	0	0	4	4	
2	1	1	2	4	3	6	5	0
3	1	2	5	6	4	11	9	0
4	6	8	4	11	3	15	7	0
5	3	11	1	15	4	16	5	0
6	7	18	5	18	0	23	5	2
7	5	23	4	23	0	27	4	0
8	2	25	1	27	2	28	3	0
9	4	29	4	29	0	33	4	1
10	1	30	3	33	3	36	6	0
11	4	34	5	36	2	41	7	0
12	4	38	3	41	3	44	6	0
13	7	45	4	45	0	49	4	1
14	6	51	5	51	0	56	5	2
15	3	54	3	56	2	59	5	0
16	8	62	2	62	0	64	2	3
17	8	70	4	70	0	74	4	6
18	2	72	3	74	2	77	5	0
19	7	79	1	79	0	80	1	2
20	4	83	2	83	0	85	2	3
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
100	5	415	2	416	1	418	3	0
Total	415		317		174		491	101

Analysis

$$\begin{aligned}\text{Average waiting time} &= \frac{\text{total time customers wait in queue (minutes)}}{\text{total numbers of customers}} \\ &= \frac{174}{100} = 1.74\end{aligned}$$

$$\text{Probability (wait)} = \frac{\text{number of customers who wait}}{\text{total numbers of customers}} = \frac{54}{100} = 0.54$$

$$\text{Probability of Idle Server} = \frac{\text{Total idle time of server}}{\text{total run time of simulation}} = \frac{101}{418} = 0.24$$

$$\text{Average Service Time} = \frac{\text{Total Service Time}}{\text{Total Number of customers}} = \frac{317}{100} = 3.17$$

Analysis

$$\begin{aligned}\text{Average Time between arrivals} &= \frac{\text{sum of all times between arrivals}}{\text{number of arrivals} - 1} \\ &= \frac{415}{99} = 4.19\end{aligned}$$

$$\begin{aligned}\text{Average waiting time of those who wait} &= \frac{\text{total time customers wait in queue}}{\text{total number of customers that wait}} \\ &= \frac{174}{54} = 3.22\end{aligned}$$

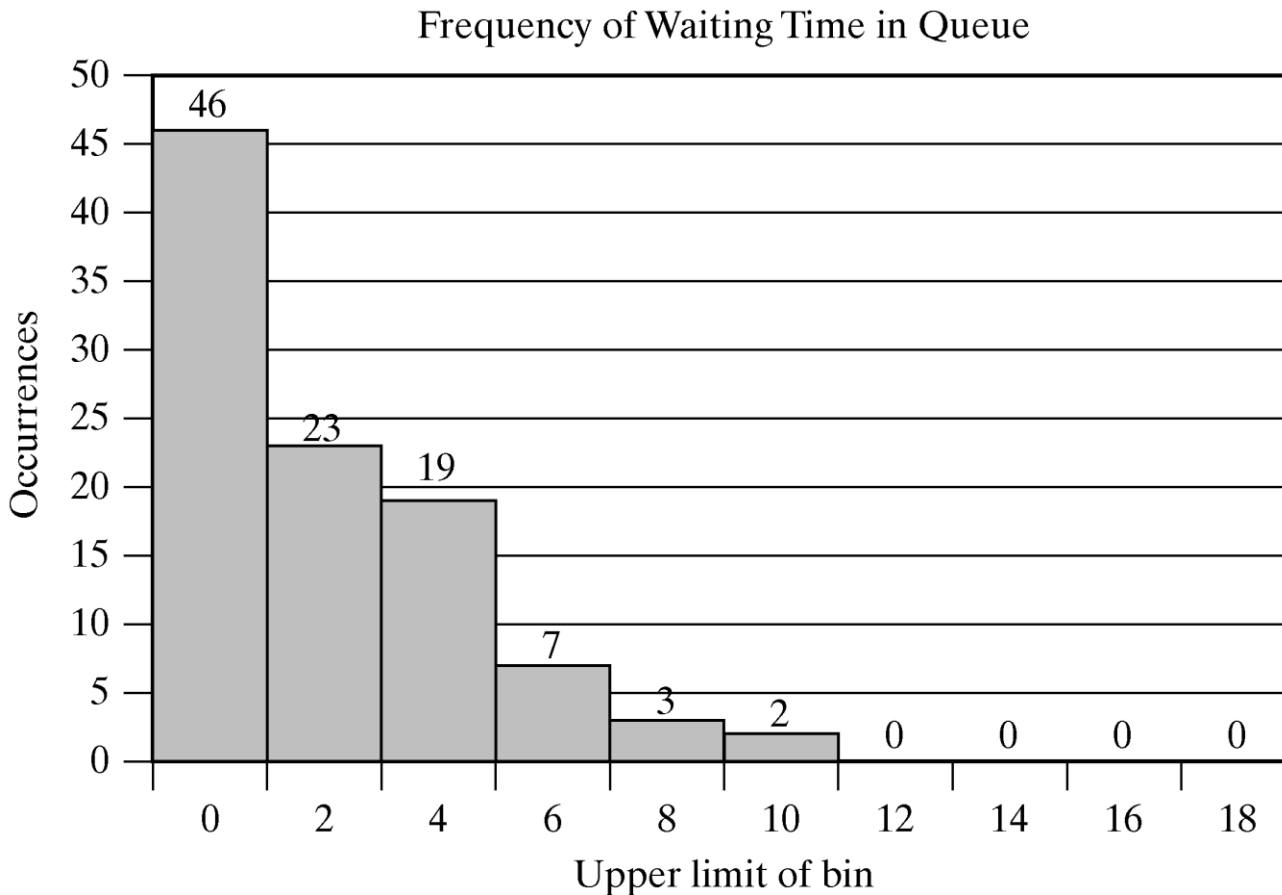
$$\begin{aligned}\text{Average time customer spends in the system} &= \frac{\text{total time customers spend in the system}}{\text{total number of customers}} \\ &= \frac{491}{100} = 4.91\end{aligned}$$

Or

$$\begin{aligned}\text{Average time customer spends in the system} &= \text{average waiting time} + \text{average service time} \\ &= 1.74 + 3.17 = 4.91 \text{ minutes}\end{aligned}$$

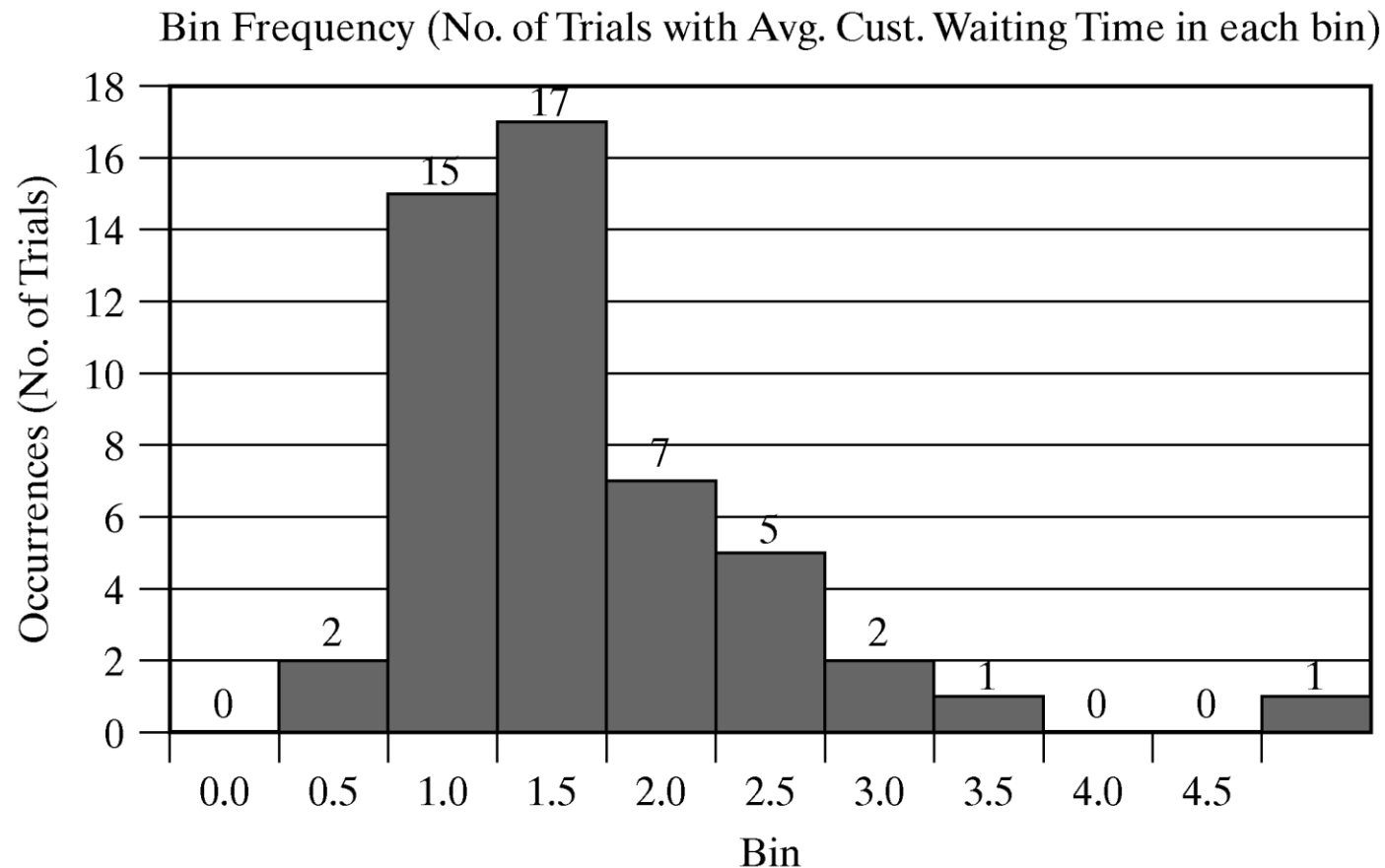
Waiting Time Distribution

- Waiting time in queue for the **first trial** of 100 customers is shown below



Average Waiting Time

- Average waiting time **over 50 trials** is = 1.5
- Histogram of 50 average waiting times over 50 trials is shown below



Able-Baker Call Center Problem (Example 2.6)

- Two Service Channels (or Servers)
- A computer technical support center where personnel take calls and provide service. The time between calls ranges from 1 to 4 minutes, with distribution as shown in Table 2.12.
- There are two technical support people-Able and Baker. Able is more experienced and can provide service faster than Baker.
- The distributions of their service times are shown in Tables 2.13 and 2.14
- **Policy:** A simplifying rule is that Able gets the call if both technical support people are idle. Able has seniority.
- The solution would be different if the decision were made at random or by any other rule
- How well this policy is working??

Distributions

Table 2.11 Interarrival Distribution of Calls for Technical Support

<i>Time between Arrivals (Minutes)</i>	<i>Probability</i>	<i>Cumulative Probability</i>	<i>Random-Digit Assignment</i>
1	0.25	0.25	01–25
2	0.40	0.65	26–65
3	0.20	0.85	66–85
4	0.15	1.00	86–00

Average=2.25

Table 2.12 Service Distribution of Able

<i>Service Time (Minutes)</i>	<i>Probability</i>	<i>Cumulative Probability</i>	<i>Random-Digit Assignment</i>
2	0.30	0.30	01–30
3	0.28	0.58	31–58
4	0.25	0.83	59–83
5	0.17	1.00	84–00

Average=3.29

Table 2.13 Service Distribution of Baker

<i>Service Time (Minutes)</i>	<i>Probability</i>	<i>Cumulative Probability</i>	<i>Random-Digit Assignment</i>
3	0.35	0.35	01–35
4	0.25	0.60	36–60
5	0.20	0.80	61–80
6	0.20	1.00	81–00

Average=4.25

Simulation Steps

- Step 1:

- For Caller k , **generate an interarrival time** A_k . Add it to the previous arrival time T_{k-1} to get the arrival time of Caller k as $T_k = T_{k-1} + A_k$.

Simulation Steps (continued ..)

■ Step 2:

- If Able is idle, Caller k begins service with Able at the **current time** T_{now}
- Able's service completion time, $T_{fin.A}$ is given by $T_{fin.A} = T_{now} + T_{svc.A}$ where $T_{svc.A}$ is the service time generated from Able's Service Time Distribution.
- Caller k 's time in system, T_{sys} , is given by $T_{sys} = T_{fin.A} - T_k$
- Because Able was idle, Caller k 's delay, T_{wait} , is given by $T_{wait} = 0$
- If Able is busy, but Baker is idle, Caller k begins service with Baker at the current time T_{now}
- Baker's service completion time, $T_{fin.B}$ is given by $T_{fin.B} = T_{now} + T_{svc.B}$ where $T_{svc.B}$ is the service time generated from Baker's Service Time Distribution.
- Caller k 's time in system, T_{sys} , is given by $T_{sys} = T_{fin.B} - T_k$
- Because Baker was idle, Caller k 's delay, T_{wait} , is given by $T_{wait} = 0$

Simulation Steps (contd ..)

■ Step 3:

- If Able and Baker are both busy, then calculate the time at which the first one becomes available, as follows: $T_{beg} = \min(T_{fin.A}, T_{fin.B})$.
- Caller k begins service at T_{beg} . When service for Caller k begins, set $T_{now} = T_{beg}$.
- Then compute $T_{fin.A}$ or $T_{fin.B}$ as in Step 2.
- Caller k 's time in system, T_{sys} , is given by $T_{sys} = T_{fin.A} - T_k$ or $T_{sys} = T_{fin.B} - T_k$, as appropriate.

Simulation Table for Call-Center Example

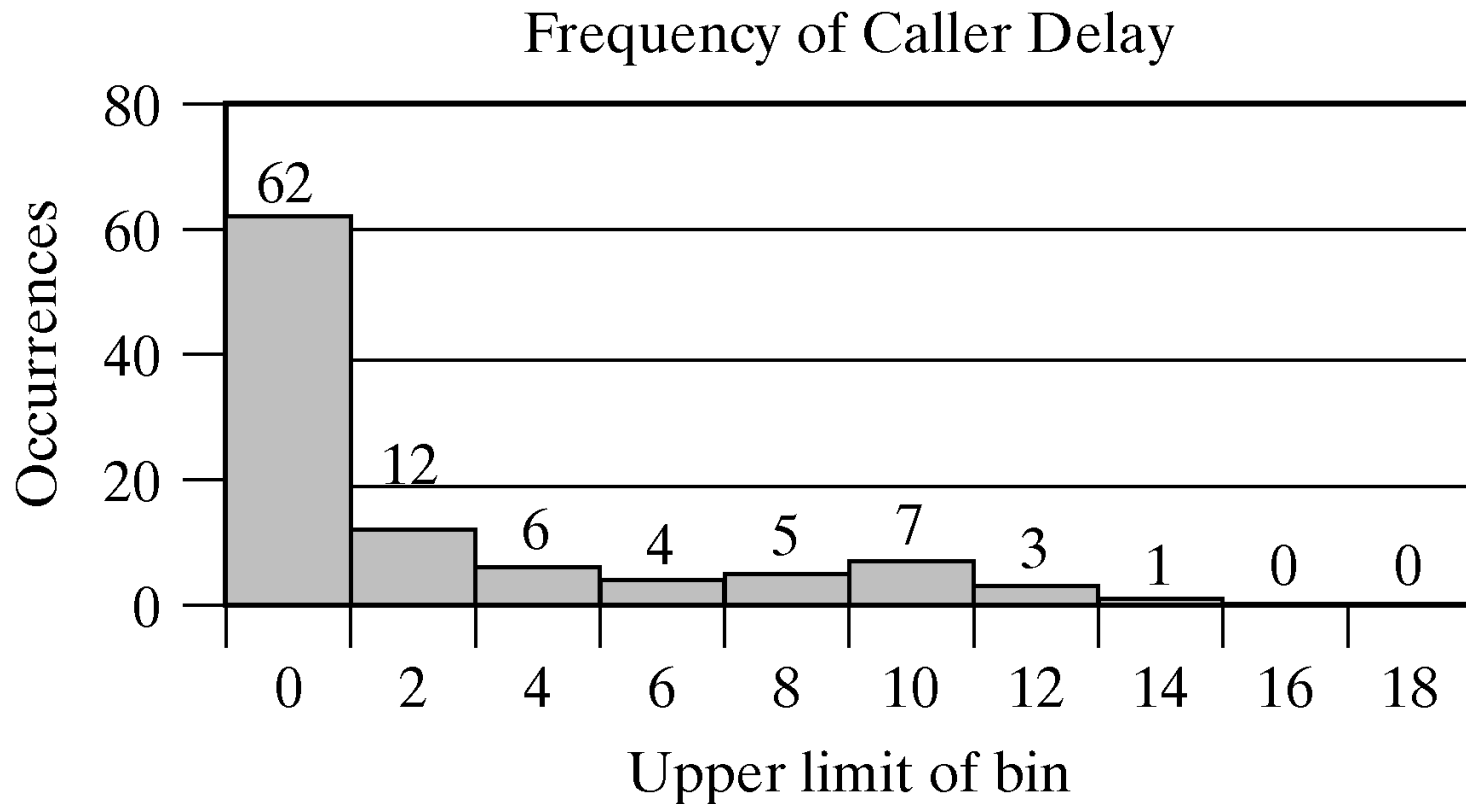
- Total Customer Delay = 211 or about 2.11 minutes per caller
- Total time in system = 564 or 5.64 minutes per caller

Table 2.14 Simulation Table for Call-Center Example

	Clock	Clock	Clock			Clock	Clock	Clock			
Caller Number	Interarrival Time (Minutes)	Arrival Time	When Able Available	When Baker Available	Server Chosen	Service Time (Minutes)	Time Service Begins	Able's Service Completion Time	Baker's Service Completion Time	Caller Delay (Minutes)	Time in System (Minutes)
1		0	0	0	Able	2	0	2		0	2
2	2	2	2	0	Able	2	2	4		0	2
3	4	6	4	0	Able	2	6	8		0	2
4	2	8	8	0	Able	4	8	12		0	4
5	1	9	12	0	Baker	3	9		12	0	3
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
100	1	219	221	219	Baker	4	219		223	0	4
Total										211	564

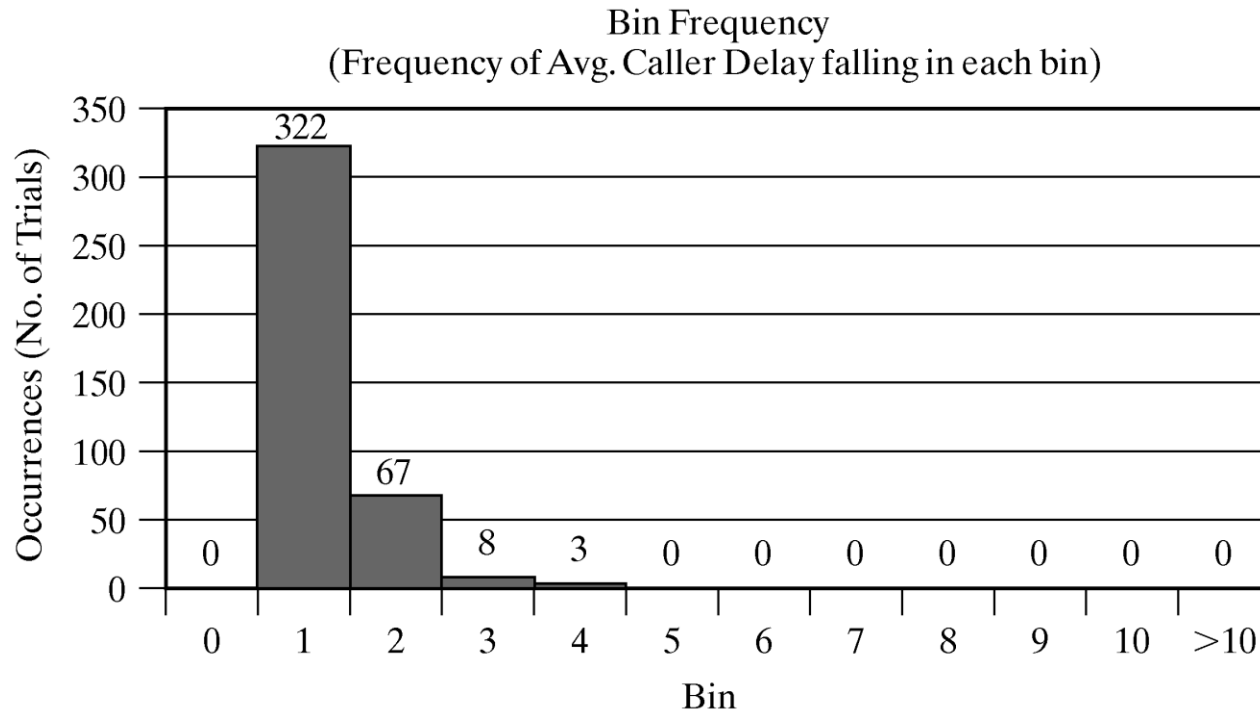
Caller Delay

- One trial with 100 callers



Average Caller Delay

- Experiment with 400 trials each with 100 callers
- 19% average delays are longer than 1 minute (78 of 400)
- only 2.75% (11 of 400) are longer than 2 minutes



(Example: Bin 2 contains all values >1 and ≤ 2)
(The right-most bin contains all values >10)

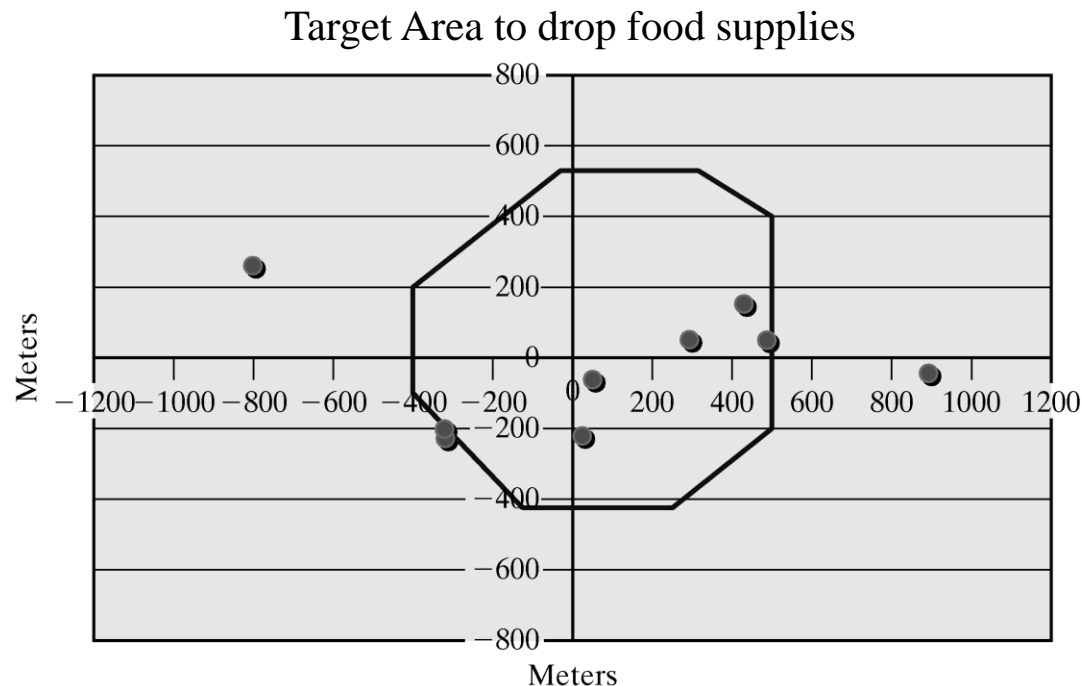
Points to ponder



- What happens if the policy is changed?
- Is it cost effective to add another server?
- Can one server handle all calls in this example?

Relief Mission (Example 2.10)

- An aircraft attempting to drop food supplies in a flood affected area, as shown in Figure. The aircraft flies in the horizontal direction and carries 10 food bags. The aiming point is (0,0).
- If a food bag falls anywhere in the target area, a hit is scored; otherwise, the it is a miss.



Relief Mission (continued ..)

- The point of impact is assumed to be *normally distributed* around the aiming point with a standard deviation of 400 meters in the direction of flight and 200 meters in the perpendicular direction
- The problem is to simulate the operation and make statements about the number of food bags that fall in the target area.
- Recall that the *standardized normal variate*, Z , having *mean* 0 and *standard deviation* 1 is distributed as $Z=(X-\mu)/\sigma$, where X is a normal random variable, μ is the mean of distribution X and σ is the standard deviation of X
- Then $X=Z\sigma_X$ and $Y=Z\sigma_Y$ where (X,Y) are the simulated co-ordinates of the food bag after it has fallen (*What is the Mean???*)
- With $\sigma_X=400$ and $\sigma_Y=200$ we have $X=400.Z_i$ and $Y=200.Z_j$
- The i and j subscripts have been added to indicate that the values of Z should be different. (*Why???*)
- We will see later how to generate Normal Random Variables

Relief Mission (continued ..)

- RNN_x (RNN_y) stands for "random normal number to compute the X (Y) coordinate" and corresponds to Z_i (Z_j)

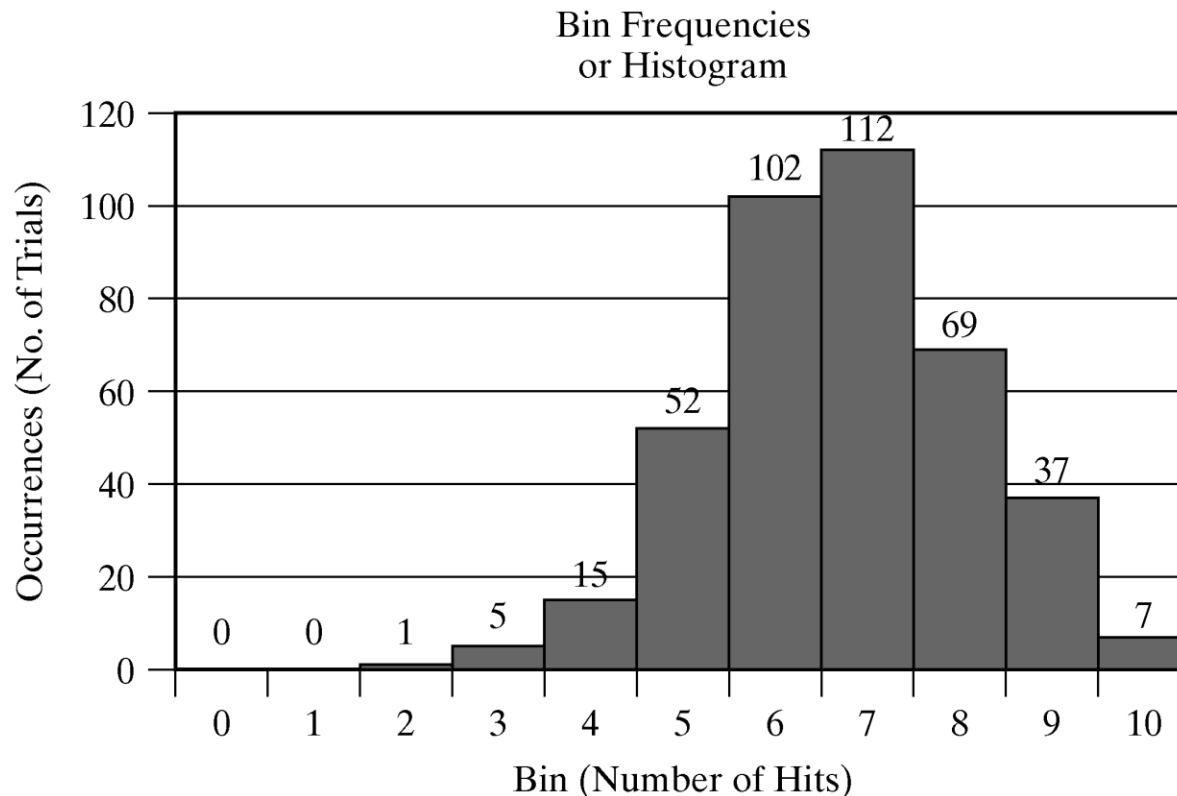
Table 2.26 Simulated Run to drop food supplies

Bag#	RNN_x	X Coordinate ($400 RNN_x$)	RNN_y	Y Coordinate ($200 RNN_y$)	Result ^a
1	2.2296	891.8	-0.1932	-38.6	Miss
2	-2.0035	-801.4	1.3034	260.7	Miss
3	-3.1432	-1257.3	0.3286	65.7	Miss
4	-0.7968	-318.7	-1.1417	-228.3	Miss
5	1.0741	429.6	0.7612	152.2	Hit
6	0.1265	50.6	-0.3098	-62.0	Hit
7	0.0611	24.5	-1.1066	-221.3	Hit
8	1.2182	487.3	0.2487	49.7	Hit
9	-0.8026	-321.0	-1.0098	-202.0	Miss
10	0.7324	-293.0	0.2552	51.0	Hit

^aTotal: 5 hits, 5 misses

Results

- An experiment was run with 400 trials (each trial 10 food bags)
- The results range from two hits to ten hits. Average is 6.72 hits
- 44% $[(175/400) \times 100\%]$ of the relief mission runs there are six or fewer hits. In about 71% of the cases, $[(283/400) \times 100\%]$ there were six, seven, or eight hits



Spreadsheets

- *Excel Spreadsheets are available for these three examples on the connex*
- *Understand the functionality, experiment*