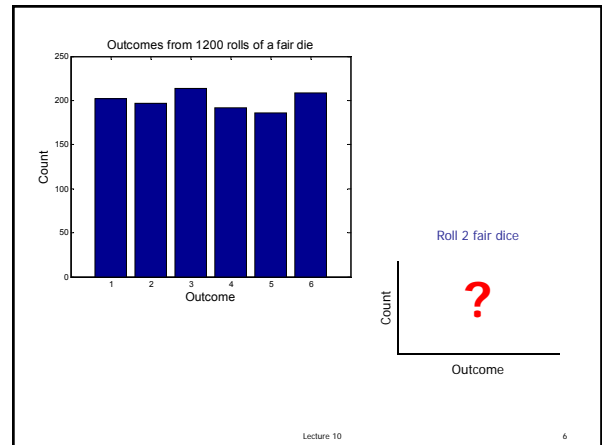


- Previous Lecture:
  - Executing a user-defined function
  - Function scope
- Today's Lecture:
  - Subfunction
  - 1-d array—vector
  - Probability and random numbers
  - Simulation using random numbers, vectors
- Announcement:
  - Project 3 due Monday 10/5 at 11pm

- Execute the statement `y = foo(x)`
- Matlab looks for a function called `foo` (m-file called `foo.m`)
  - Argument (value of `x`) is copied into function `foo`'s **local parameter**
    - called "pass-by-value," one of several argument passing schemes used by programming languages
  - Function code executes **within its own workspace**
  - At the end, the function's **output argument** (value) is sent from the function to the place that calls the function. E.g., the value is assigned to `y`.
  - Function's **workspace is deleted**
    - If `foo` is called again, it starts with a new, empty workspace

- ### Subfunction
- There can be more than one function in an M-file
  - **top** function is the main function and has the name of the file
  - remaining functions are **subfunctions, accessible only by the functions in the same m-file**
  - Each (sub)function in the file begins with a **function header**
  - Keyword `end` is not necessary at the end of a (sub)function



- ### 1-d array: vector
- An array is a **named** collection of **like** data organized into rows or columns
  - A 1-d array is a row or a column, called a **vector**
  - An **index** identifies the **position** of a value in a vector
- |   |    |    |   |
|---|----|----|---|
| v | .8 | .2 | 1 |
|   | 1  | 2  | 3 |

- Here are a few different ways to create a vector
- ```
count = zeros(1,6)      count = [0 0 0 0 0 0]
```
- Similar functions: `ones`, `rand`
- ```
a = linspace(10,30,5)   a = [10 15 20 25 30]
```
- ```
b = 7:-2:0              b = [7 5 3 1]
```
- ```
c = [3 7 2 1]          c = [3 7 2 1]
```
- ```
d = [3; 7; 2]          d = [3; 7; 2]
```
- ```
e = d'                  e = [3 7 2]
```

Start with drawing a single line segment

```

a= 0; % x-coord of pt 1
b= 1; % y-coord of pt 1
c= 5; % x-coord of pt 2
d= 3; % y-coord of pt 2
plot([a c], [b d], '-*')
    
```

x-values (a vector)      y-values (a vector)      Line/marker format

Lecture 10 9

Making an x-y plot

```

a= [0 4 3 8]; % x-coords
b= [1 2 5 3]; % y-coords
plot(a, b, '-*')
    
```

x-values (a vector)      y-values (a vector)      Line/marker format

Lecture 10 10

Making an x-y plot with multiple graphs (lines)

```

a= [0 4 5 8];
b= [1 2 5 3];
f= [0 4 6 8 10];
g= [2 2 6 4 3];
plot(a,b,'-*',f,g,'c')
legend('graph 1 name', 'graph 2 name')
xlabel('x values')
ylabel('y values')
title('My graphs', 'FontSize',14)
    
```

Lecture 10 11

Array index starts at 1

x	5	.4	.91	-4	-1	7
	1	2	3	4	5	6

Let  $k$  be the index of vector  $x$ , then

- $k$  must be a positive integer
- $1 \leq k \leq \text{length}(x)$
- To access the  $k^{\text{th}}$  element:  $x(k)$

Lecture 10 14

Accessing values in a vector

score	93	99	87	80	85	82
	1	2	3	4	5	6

Given the vector **score** ...

```

score(4)= 80;
score(5)= (score(4)+score(5))/2;
k= 1;
score(k+1)= 99;
    
```

See `plotComparison2.m`

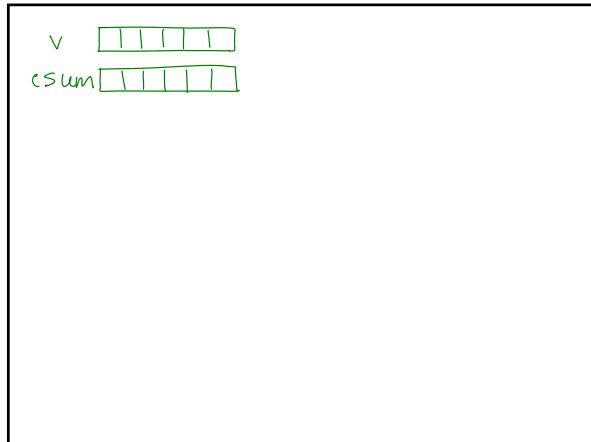
Lecture 10 16

Example

- Write a program fragment that calculates the cumulative sums of a given vector  $v$ .
- The cumulative sums should be stored in a vector of the same length as  $v$ .

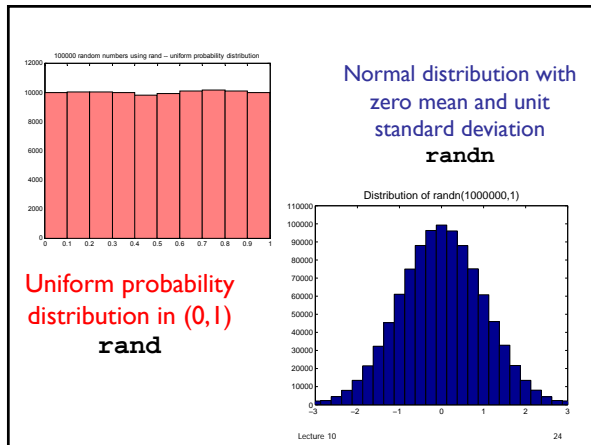
1, 3, 5, 0     $v$   
 1, 4, 9, 9    cumulative sums of  $v$

Lecture 10 17



Random numbers

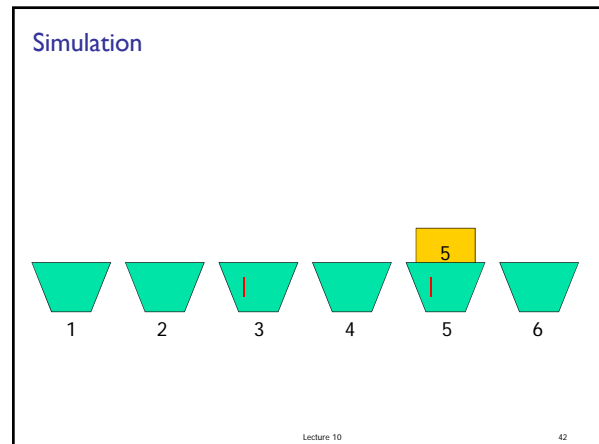
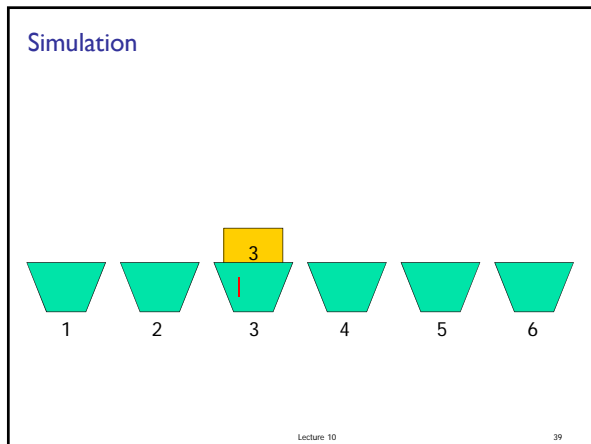
- *Pseudorandom* numbers in programming
- Function `rand(...)` generates random real numbers in the interval (0,1). All numbers in the interval (0,1) are equally likely to occur—**uniform** probability distribution.
- Examples:
  - `rand(1)` one random # in (0,1)
  - `6*rand(1)` one random # in (0,6)
  - `6*rand(1)+1` one random # in (1,7)

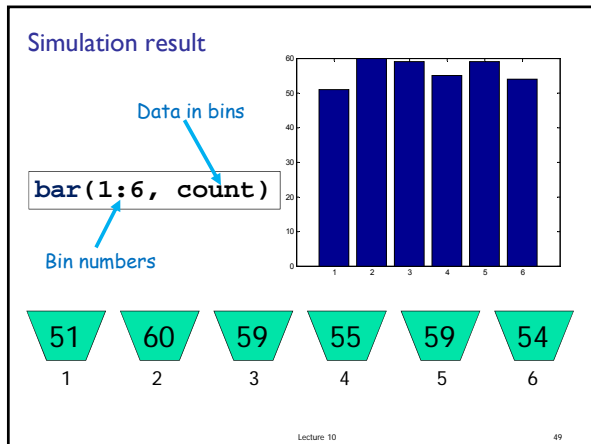


Simulate a fair 6-sided die

Which expression(s) below will give a random *integer* in [1..6] with equal likelihood?

- A `round(rand*6)`
- B `ceil(rand*6)`
- C Both expressions above





Keep tally on repeated rolls of a fair die

Repeat the following:

```
% roll the die
% increment correct "bin"
```

Lecture 10 50

```
function count = rollDie(rolls)
    FACES= 6; % #faces on die
    count= zeros(1,FACES);
    % Count outcomes of rolling a FAIR die
    for k= 1:rolls
        % Roll the die
        % Increment the appropriate bin
    end
    % Show histogram of outcome
```

Lecture 10 51

```
% Count outcomes of rolling a FAIR die
count= zeros(1,6);
for k= 1:100
    face= ceil(rand*6);
    if face==1
        count(1)= count(1) + 1;
    elseif face==2
        count(2)= count(2) + 1;
    :
    elseif face==5
        count(5)= count(5) + 1;
    else
        count(6)= count(6) + 1;
    end
end
```

Lecture 10 50

```
function count = rollDie(rolls)
    FACES= 6; % #faces on die
    count= zeros(1,FACES);
    % Count outcomes of rolling a FAIR die
    for k= 1:rolls
        % Roll the die
        face= ceil(rand*FACES);
        % Increment the appropriate bin
        count(face)= count(face) + 1;
    end
    % Show histogram of outcome
```

Lecture 10 56

% Simulate the rolling of 2 fair dice  
totalOutcome= ???

- A ceil(rand\*12)
- B ceil(rand\*11)+1
- C floor(rand\*11)+2
- D 2 of the above
- E None of the above

Lecture 10 59