

---

# Monte-Carlo Experiments

**ECE 341 Random Processes**

**Lecture #2**

Please visit Bison Academy for course syllabus, lecture notes,  
recorded lectures, homework sets, and solutions

[www.BisonAcademy.com](http://www.BisonAcademy.com)

---

# Introduction

The probability of an event is defined as

*The number of times an event happens  
as the number of trials goes to infinity.*

This leads to one way to compute the probability of an event:

- Write a program to play a game one time
- Then play the game one million times
- Count the number of times the event happens

The probability of the event happening is then approximately

- The number of times the event happened,
- Divided by 1 million

This is termed *A Monte Carlo Experiment*

WIKIPEDIA  
The Free Encyclopedia

Q ∞ ⋮

≡ Monte Carlo method

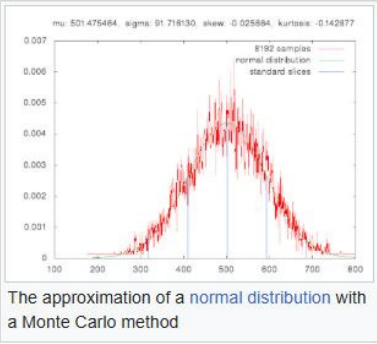
53 languages

Article Talk

Tools

From Wikipedia, the free encyclopedia

Not to be confused with [Monte Carlo algorithm](#).



The approximation of a normal distribution with a Monte Carlo method

**Monte Carlo methods**, or **Monte Carlo experiments**, are a broad class of [computational algorithms](#) that rely on [repeated random sampling](#) to obtain numerical results. The underlying concept is to use [randomness](#) to solve problems that might be [deterministic](#) in principle. The name comes from the [Monte Carlo Casino](#) in Monaco, where the primary developer of the method, mathematician [Stanislaw Ulam](#), was inspired by his uncle's gambling habits.

Monte Carlo methods are mainly used in three distinct problem classes: optimization, numerical integration, and generating draws from a probability distribution. They can also be used to model phenomena with significant uncertainty in inputs, such as calculating the risk of a nuclear power plant failure. Monte Carlo methods are often implemented using computer simulations, and they can provide

# Matlab

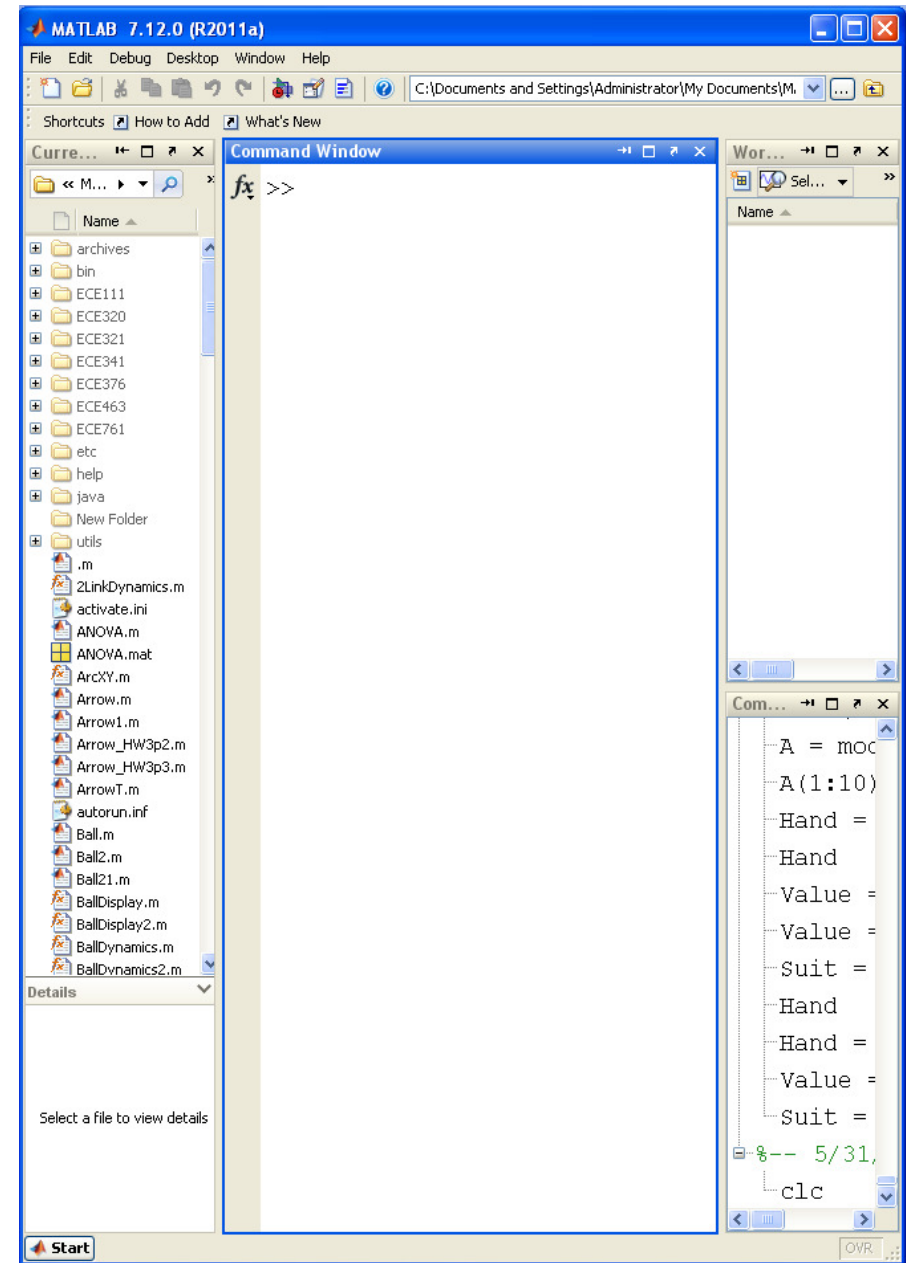
- A little background on Matlab

Matlab is a computer program we'll use throughout this course

- It's essentially a calculator
- It's also a programming language

The default screen includes

- Current folder
  - Current directory - close
- Command Window
  - Keep
- Workspace
  - List of variables - close
- Command History
  - Don't care - close



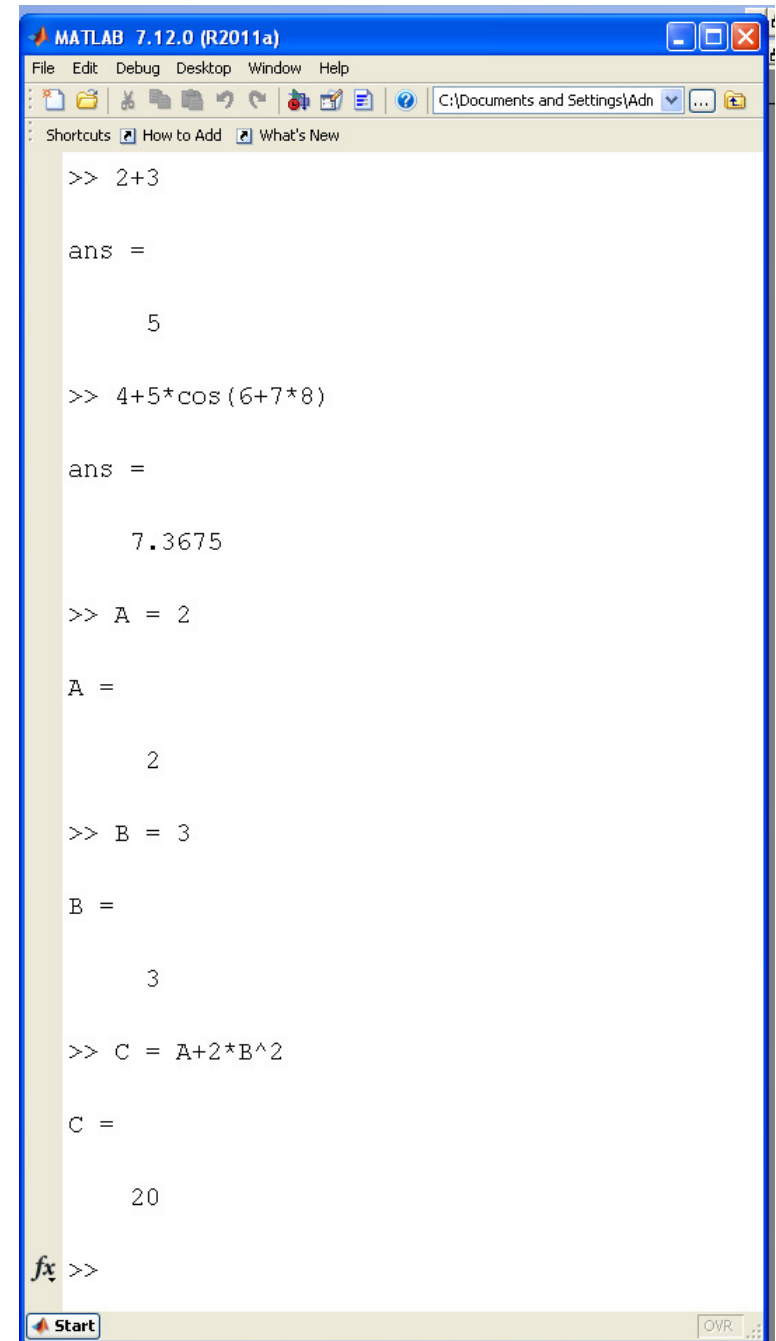
# Matlab as a Calculator

The command window acts like a calculator

- You can type in operations
- It then evaluates these operations

You can assign numbers to variables

- Case sensitive
- Variables can be functions of variables
- Valid syntax
  - Variable name
  - Equals
  - Operation to evaluate



```
MATLAB 7.12.0 (R2011a)
File Edit Debug Desktop Window Help
C:\Documents and Settings\Adn
Shortcuts How to Add What's New

>> 2+3

ans =

    5

>> 4+5*cos(6+7*8)

ans =

    7.3675

>> A = 2

A =

    2

>> B = 3

B =

    3

>> C = A+2*B^2

C =

    20

fx >>
```

# Matlab is a Matrix Language

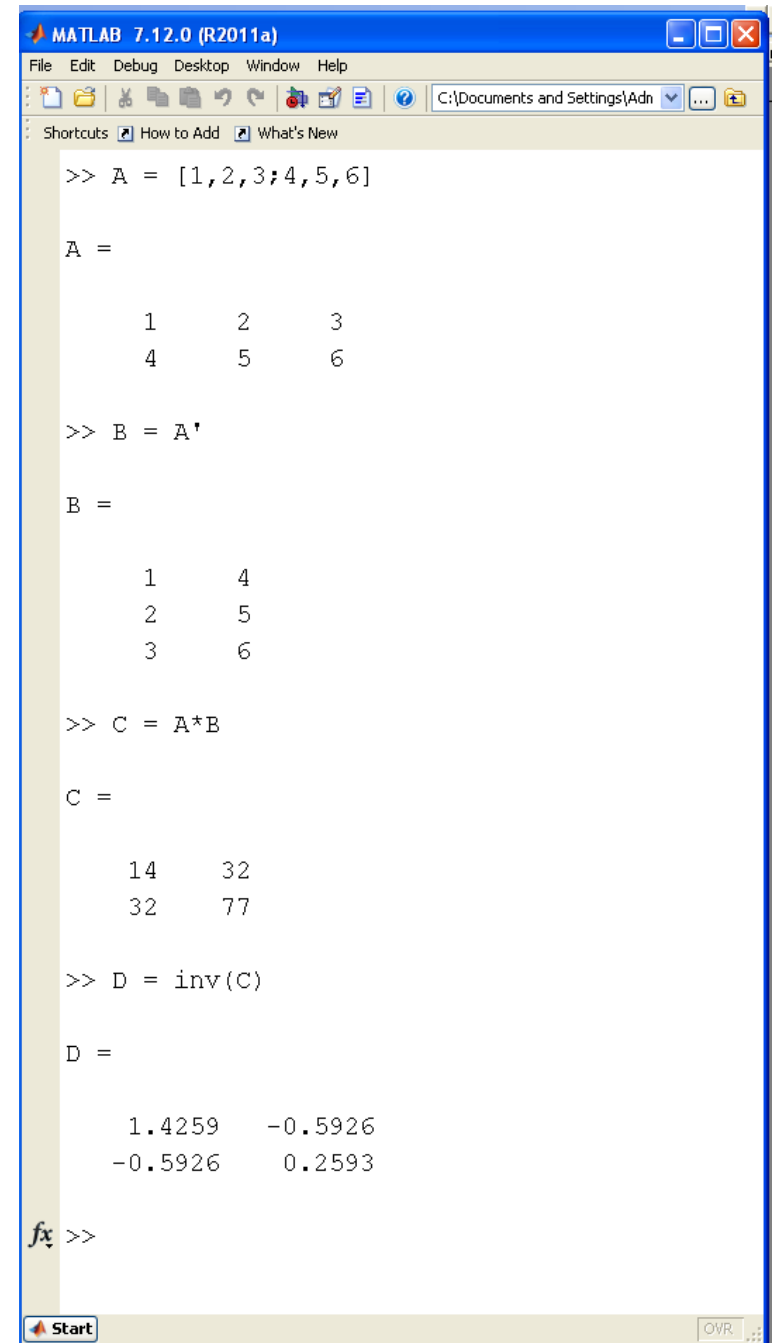
Matlab can be treated like a calculator that works with matrices

[        start of matrix  
]        end of matrix  
,        next column  
;        next row

+        addition  
-        subtraction  
\*        multiplication  
/        division  
'        transpose

.\*       element-by-element multiply  
./       element-by-element division  
.^       element raise to a power

inv(A)   matrix inverse



```
MATLAB 7.12.0 (R2011a)
File Edit Debug Desktop Window Help
C:\Documents and Settings\Adn
Shortcuts How to Add What's New

>> A = [1,2,3;4,5,6]

A =

     1     2     3
     4     5     6

>> B = A'

B =

     1     4
     2     5
     3     6

>> C = A*B

C =

    14    32
    32    77

>> D = inv(C)

D =

    1.4259   -0.5926
   -0.5926    0.2593

fx >>
```

# Flow Control

Matlab is a programming language with loops

## For-Loop

```
for i=1:10
    t = t + dt;
end
```

## While-Loop

```
time = 0;
while(time < 10)
    x = x + dx*dt;
    t = t + dt;
end
```

## If

```
if(time < 10)
    x = 0;
end
```

## If-Else

```
if(x>y)
    points = points + 1;
elseif(x == y)
    points = points + 0.5;
else
    points = points + 0;
end
```

### Command Window

```
>> X = zeros(1,4);
>> for i=1:4
        X(i) = i*i;
    end
>> X

X =

         1         4         9        16

>> x = 10;
>> dx = 0;
>> t = 0;
>> dt = 0.01;
>> while(x>0)
        ddx = -9.8;
        dx = dx + ddx*dt;
        x = x + dx*dt;
        t = t + dt;
    end
>> t

t =

    1.4300
```

*fx*

# Matlab Scripts

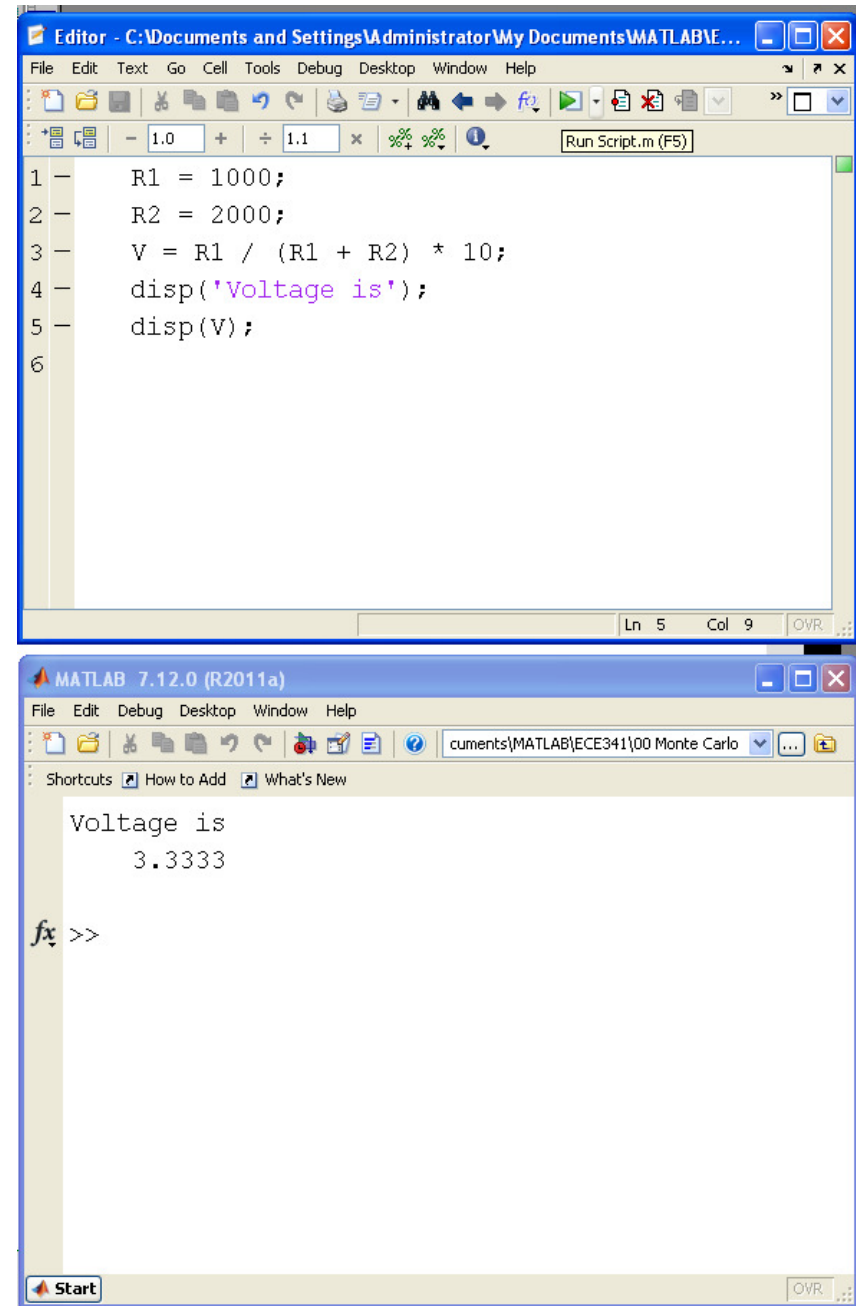
- File - New - Script
- control N

If you're going to run the same code over and over, you can place it in a script.

Each time you execute the script, it's like pasting that code into the command window.

note: This is a convenient way to build a more complex program.

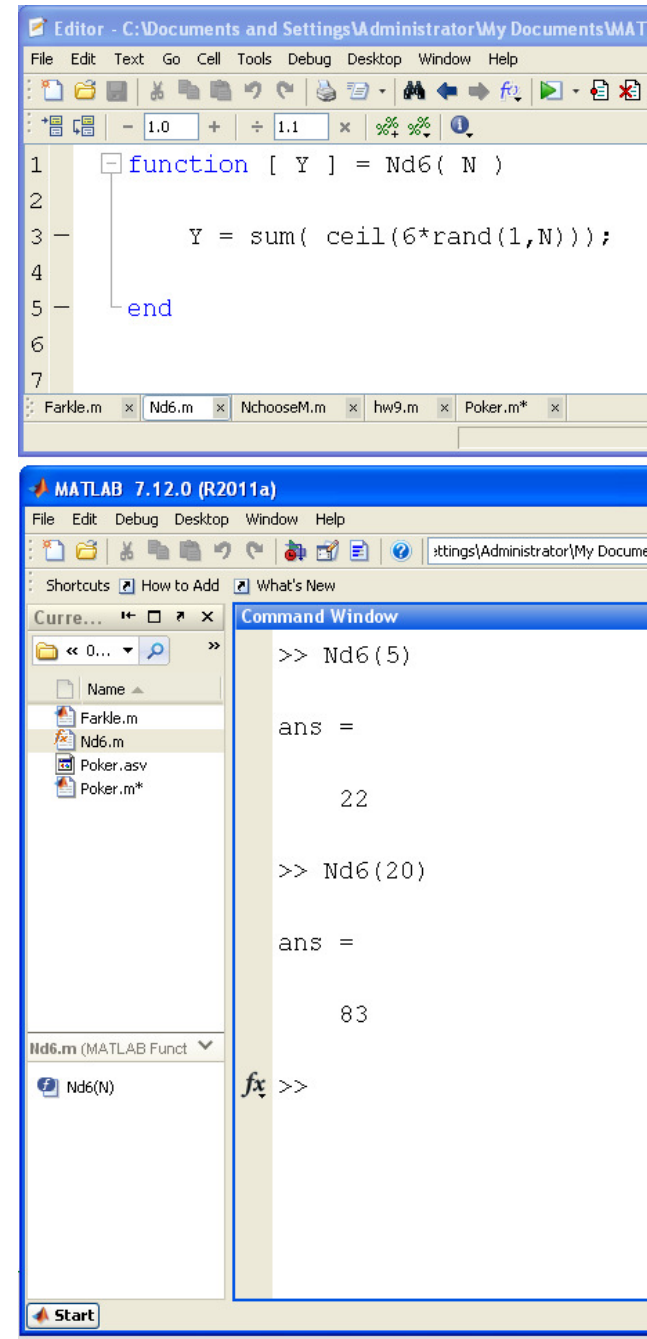
- Display the data when writing the program to see what is happening
- You can fix errors along the way



# Matlab Functions

Part of what makes Matlab so powerful is you can create your own functions

- These functions become Matlab commands that other functions can use.
- As companies build up their library of Matlab functions, they get better and better at designing their product.
- The Matlab functions become company proprietary information (design secrets).



# Random Numbers in Matlab

- `rand(4,1)`: generate a 4x1 matrix of random numbers in the range of (0,1)
- `randn(4,1)` generate a 4x1 matrix of normally distributed random numbers
- `ceil(6 * rand(4,1) )`: generate four 6-sided dice (4d6)

```
Command Window
>> rand(4,1)

ans =

    0.1712
    0.7060
    0.0318
    0.2769

>> randn(4,1)

ans =

   -0.8095
   -2.9443
    1.4384
    0.3252

>> ceil(6 * rand(4,1))

ans =

     2
     6
     1
     3

fx >> |
```

# Useful Matlab Commands

command	Description
<code>+ - * /</code>	add, subtract, multiply, divide
<code>^</code>	raise to a power
<code>.^ .* ./</code>	element-by-element operations
<code>ceil(2.3)</code>	round up
<code>floor(2.3)</code>	round down
<code>round(2.3)</code>	round
<code>mod(x,10)</code>	x modulus 10
<code>A'</code>	matrix transpose
<code>inv(A)</code>	matrix inverse
<code>sum(A)</code>	sum of all elements
<code>max(A)</code>	maximum element of A
<code>min(A)</code>	minimum element of A
<code>pause(0.1)</code>	pause 0.1 second

Matlab Command	Description
<code>x = [0:10];</code>	create a row matrix starting from 0 going to 10, step size = 1
<code>x = [0:0.01:10];</code>	go from 0 to 10 step size of 0.01
<code>length(x)</code>	length of matrix x
<code>size(x)</code>	dimensions of matrix x
<code>rand</code>	random number in (0,1)
<code>rand(100,1)</code>	100x1 matrix of random numbers
<code>randn</code>	random with a normal distribution
<code>mean(x)</code>	mean, average, 1st moment
<code>std(x)</code>	standard deviation of x
<code>var(x)</code>	variance of x
<code>disp(A)</code>	display A
<code>tic</code>	start recording time
<code>toc</code>	display time since tic

# Monte-Carlo Experiments

- Named after Monte Carlo casino

One way to determine the probabilities

- Write a script to play a game one time
- Next, play the game N times
  - N is large, like a million
- Count how many times an event happens
  - Player A wins

The probability of winning is

- The number of events
- Divided by the number of games
- (approximately)

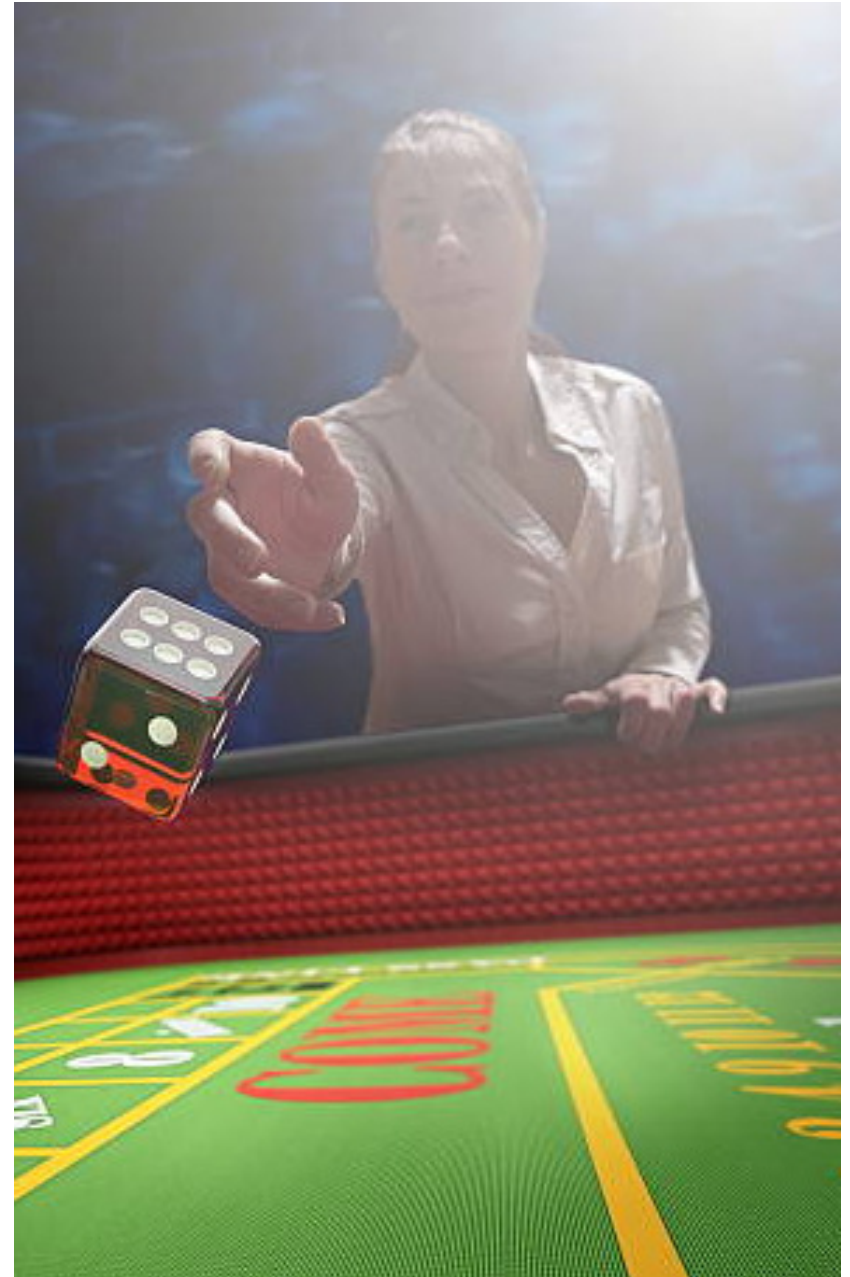


## Case 1: 6-Sided Die

- What is the probability of rolling a 1 on a 6 sided die?

Start with something simple

- We know what the result should be
- $p = 1/6$



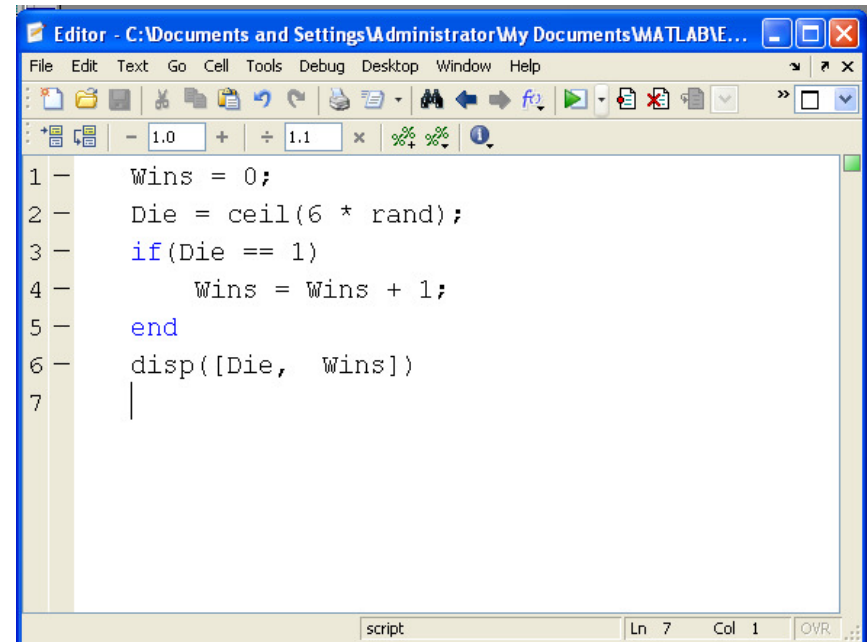
# Case 1 (cont'd)

## Monte-Carlo Solution:

- Start with playing a game one time
- Script Window (upper image)

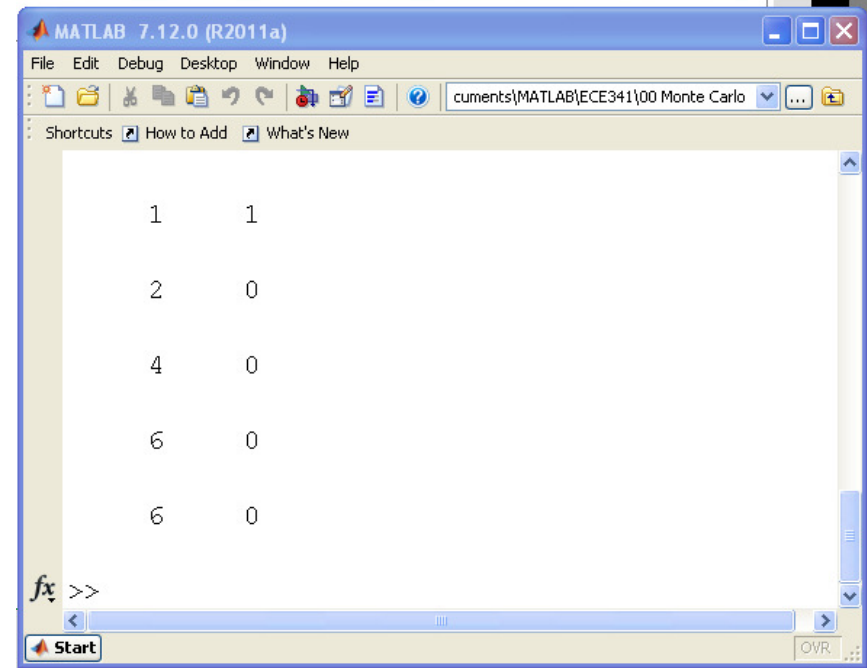
Check that the program is working properly

- Command window (lower image)
- Each time you press *run*
  - you get a different result
  - it's random
- When you roll a 1, you win



The image shows the MATLAB Editor window with a script file named 'script'. The script contains the following code:

```
1 Wins = 0;
2 Die = ceil(6 * rand);
3 if(Die == 1)
4     Wins = Wins + 1;
5 end
6 disp([Die, Wins])
7
```



The image shows the MATLAB Command Window displaying the output of the script. The output consists of five rows, each showing a die roll and the corresponding number of wins:

1	1
2	0
4	0
6	0
6	0

## Case 1 (cont'd)

Once you can play a game one time,  
play it a million times

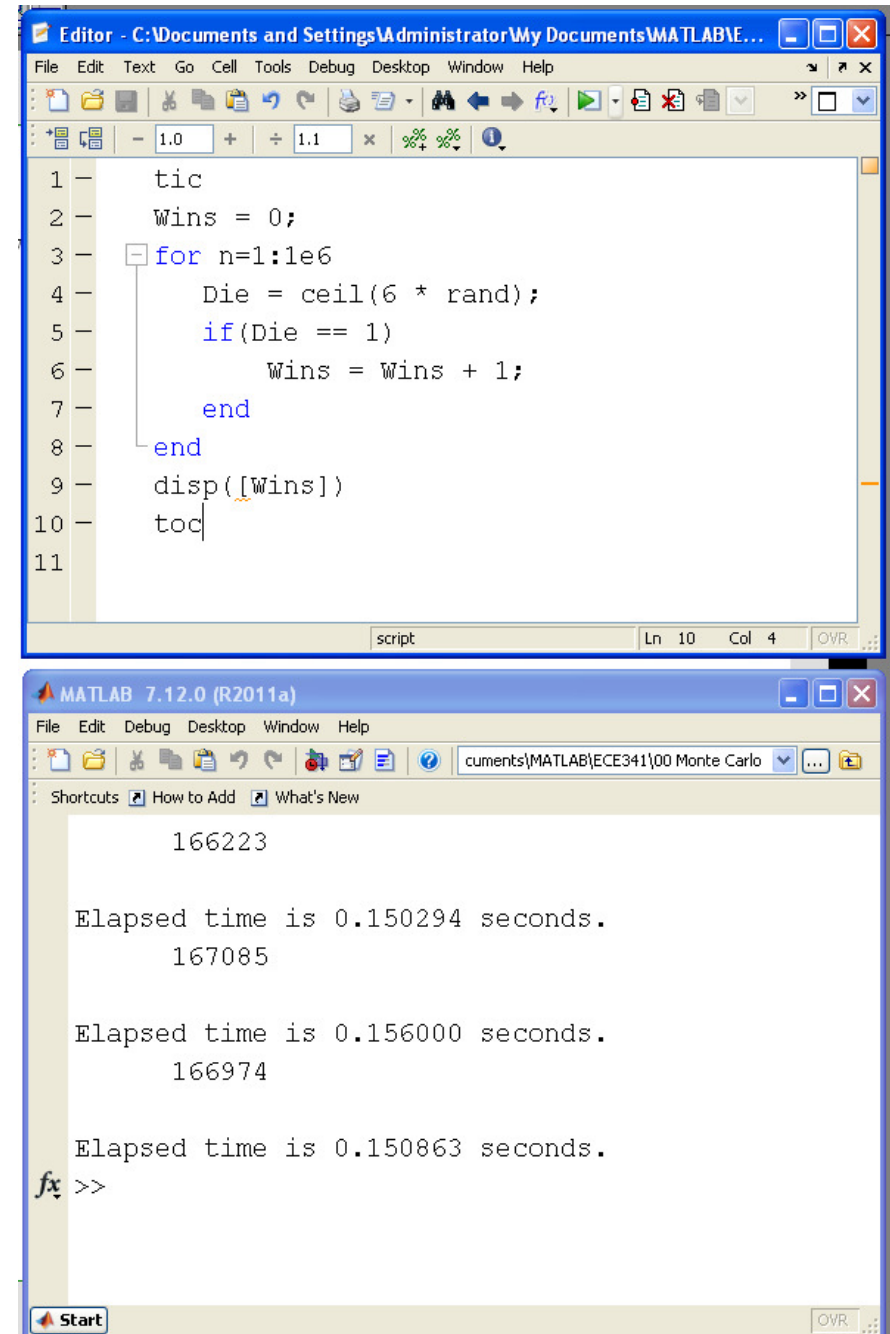
- Place previous code in a for-loop
- Count how many times you win

### Result

- Roll the die 1 million times
- Number of games won was
  - 166223, 167085, 166974
- $p \approx 0.166$

### Note:

- Results vary each trial (it's random)
- The result is approximate
- The variation tells us something
  - Future topic: t-Test



The top screenshot shows a MATLAB script editor window titled 'Editor - C:\Documents and Settings\Administrator\My Documents\MATLAB\...'. The script contains the following code:

```
1 - tic
2 - Wins = 0;
3 - for n=1:1e6
4 -     Die = ceil(6 * rand);
5 -     if(Die == 1)
6 -         Wins = Wins + 1;
7 -     end
8 - end
9 - disp([Wins])
10 - toc
11
```

The bottom screenshot shows the MATLAB 7.12.0 (R2011a) command window. It displays the results of three trials, each showing the number of wins and the elapsed time:

```
166223
Elapsed time is 0.150294 seconds.
167085
Elapsed time is 0.156000 seconds.
166974
Elapsed time is 0.150863 seconds.
fx >>
```

## Case 2: $\max(d4, d6)$ vs $d6$

### Problem

- Player A rolls a d4 and a d6
  - A's score is the maximum of the two
- Player B rolls a single d6
- Highest score wins
- B wins on ties

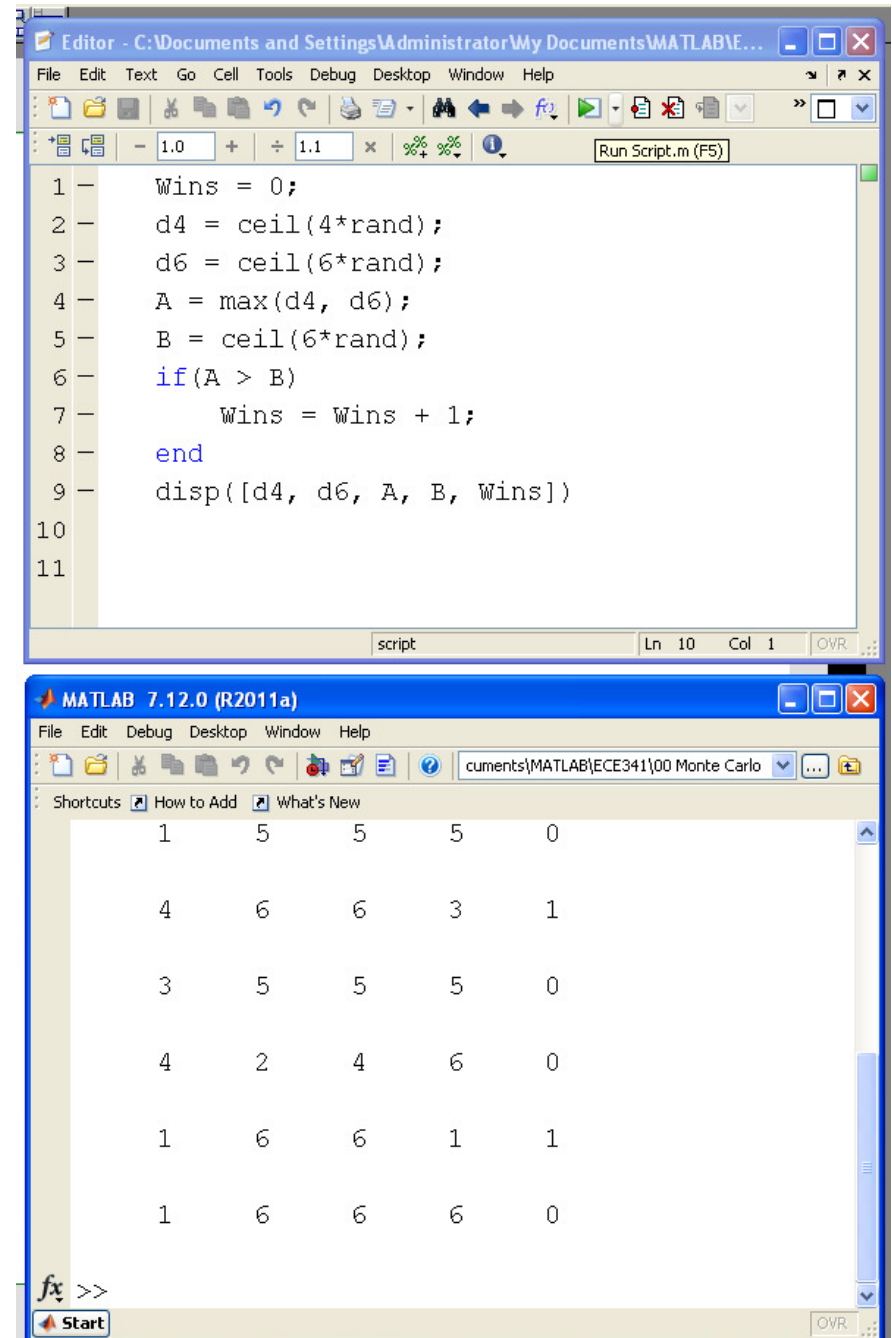
What is the probability that A will win?



## Case 2 (cont'd)

### Monte-Carlo Solution

- Start with playing the game one time
- Check that your code works
  - Col#1: d4 is random over [1,4]
  - Col#2: d6 is random over [1,6]
  - Col#3: A is the maximum of (d4, d6)
  - Col#4: B is random over [1,6]
  - Col #5: A wins when  $A > B$



The image displays two MATLAB windows. The top window is the Editor, showing a script named 'script.m' with the following code:

```
1 Wins = 0;
2 d4 = ceil(4*rand);
3 d6 = ceil(6*rand);
4 A = max(d4, d6);
5 B = ceil(6*rand);
6 if(A > B)
7     Wins = Wins + 1;
8 end
9 disp([d4, d6, A, B, Wins])
10
11
```

The bottom window is the MATLAB Command Window, showing the results of the simulation. The output is a 6x5 matrix of values:

1	5	5	5	0
4	6	6	3	1
3	5	5	5	0
4	2	4	6	0
1	6	6	1	1
1	6	6	6	0

The Command Window also shows the MATLAB prompt 'fx >>' and a 'Start' button.

## Case 2 (cont'd)

Once that works, repeat 1 million times

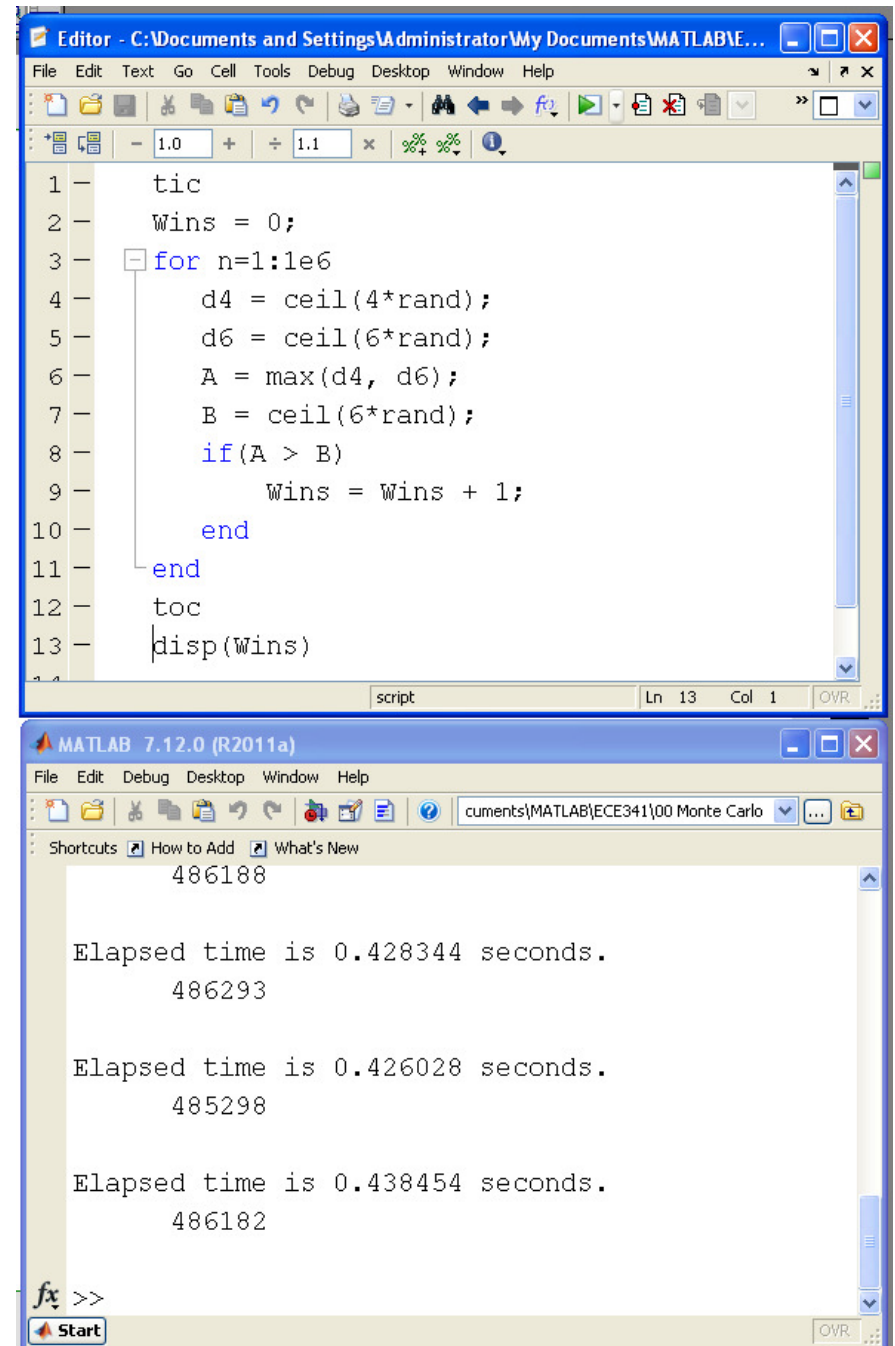
- Place previous code in a for-loop
- Count the number of successes
  - A wins

Result:

- {486188, 486293, 485298, 486182}
- About a 48.6% chance that A wins

Note

- Results are different each run
- Answers are approximate
- The variation tells us something
  - t-Test
  - Future lecture



```
Editor - C:\Documents and Settings\Administrator\My Documents\MATLABE...
File Edit Text Go Cell Tools Debug Desktop Window Help
1 - tic
2 - Wins = 0;
3 - for n=1:1e6
4 -     d4 = ceil(4*rand);
5 -     d6 = ceil(6*rand);
6 -     A = max(d4, d6);
7 -     B = ceil(6*rand);
8 -     if(A > B)
9 -         Wins = Wins + 1;
10 -    end
11 - end
12 - toc
13 - disp(Wins)
script Ln 13 Col 1 OVR
```

```
MATLAB 7.12.0 (R2011a)
File Edit Debug Desktop Window Help
Shortcuts How to Add What's New
486188

Elapsed time is 0.428344 seconds.
486293

Elapsed time is 0.426028 seconds.
485298

Elapsed time is 0.438454 seconds.
486182

fx >>
Start OVR
```

## Case 3: 5-Game Match

Assume A and B are playing a match

- Each match consists of 5 games
- A has a 60% chance of winning any given game
- Similar to basketball NBA finals

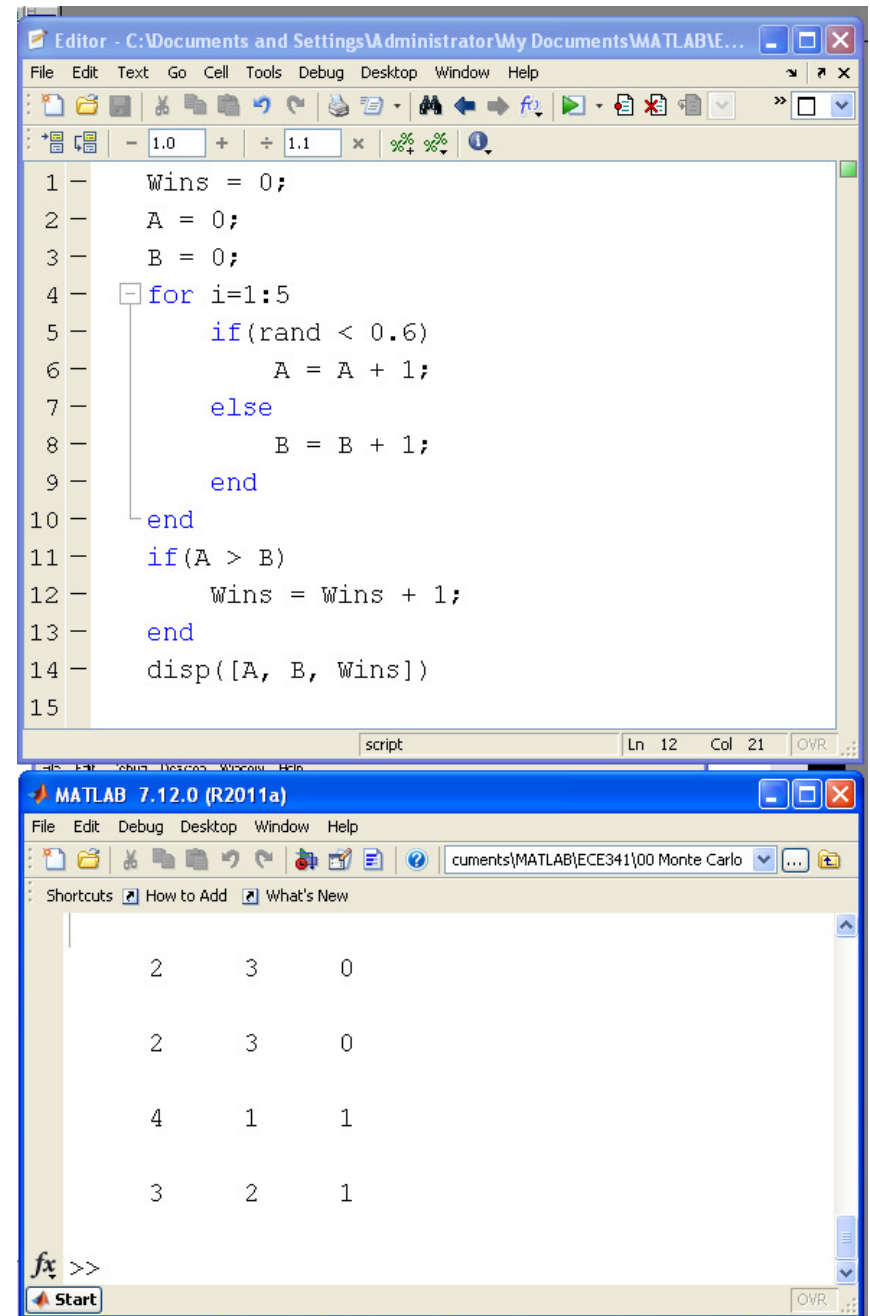
What is the probability that A will win the match?



## Case 3 (cont'd)

### Monte-Carlo Solution

- Start with playing one game
- Repeat 5 times (a match)
- Record who won the match
- Check that your code is working
  - Wins for A and B vary each match
  - $A + B = 5$  (5 game match)
  - A wins when  $A > B$



The image displays two MATLAB windows. The top window is the Editor, showing a script for a Monte Carlo simulation. The bottom window is the Command Window, showing the output of the script.

**Editor - C:\Documents and Settings\Administrator\My Documents\MATLABE...**

```
1 Wins = 0;  
2 A = 0;  
3 B = 0;  
4 for i=1:5  
5     if(rand < 0.6)  
6         A = A + 1;  
7     else  
8         B = B + 1;  
9     end  
10 end  
11 if(A > B)  
12     Wins = Wins + 1;  
13 end  
14 disp([A, B, Wins])  
15
```

**MATLAB 7.12.0 (R2011a)**

Shortcuts | How to Add | What's New

2	3	0
2	3	0
4	1	1
3	2	1

fx >>  
Start

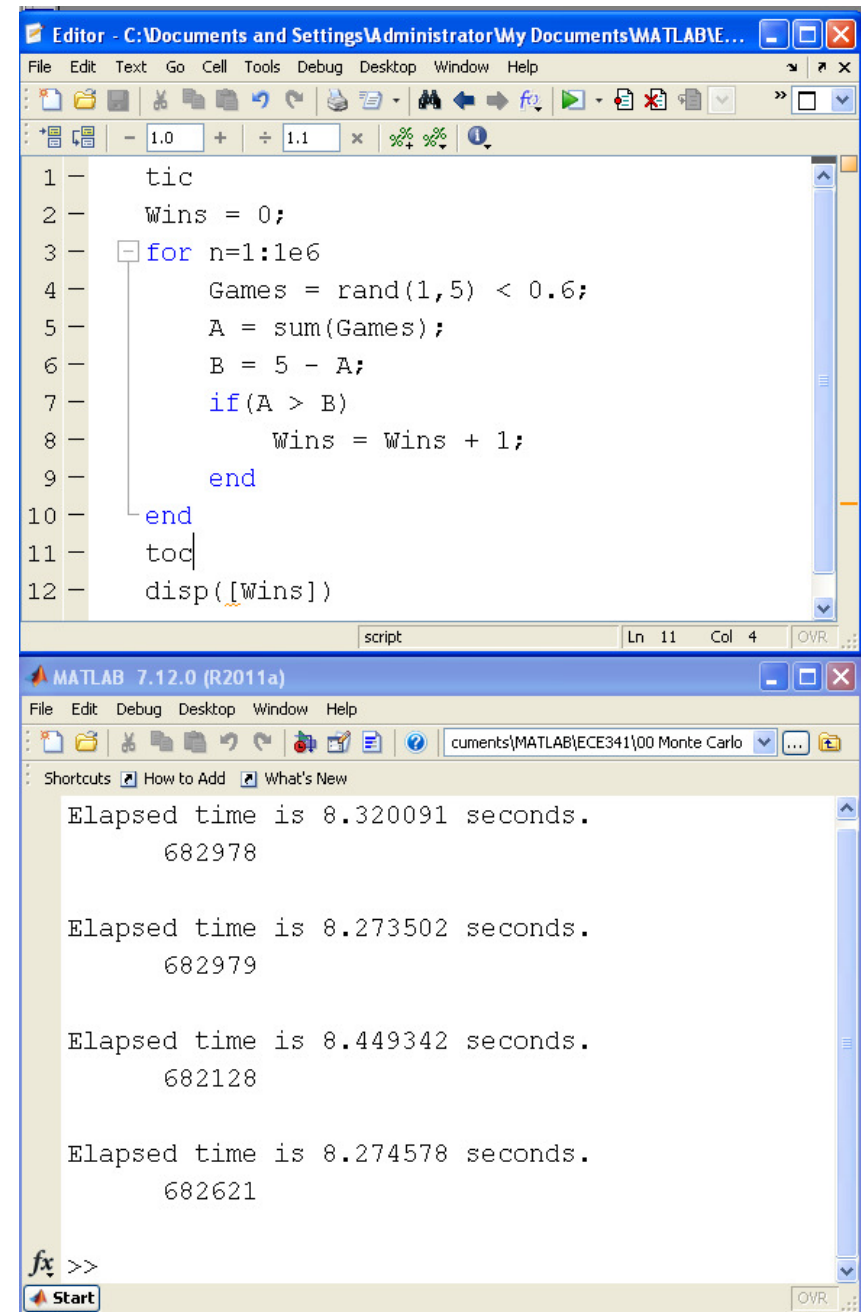
## Case 3: (cont'd)

Now repeat 1 million times

- Code modified
- Another way to play a 5-game match

Result

- A wins about 68.2% of the time



The image shows two windows from the MATLAB 7.12.0 (R2011a) environment. The top window is the MATLAB Editor, displaying a script named 'script' with the following code:

```
1 - tic
2 - Wins = 0;
3 - for n=1:1e6
4 -     Games = rand(1,5) < 0.6;
5 -     A = sum(Games);
6 -     B = 5 - A;
7 -     if(A > B)
8 -         Wins = Wins + 1;
9 -     end
10 - end
11 - toc
12 - disp([Wins])
```

The bottom window is the MATLAB Command Window, showing the output of the script. It displays four iterations of the simulation, each showing the elapsed time and the number of wins (A wins):

```
Elapsed time is 8.320091 seconds.
682978

Elapsed time is 8.273502 seconds.
682979

Elapsed time is 8.449342 seconds.
682128

Elapsed time is 8.274578 seconds.
682621
```

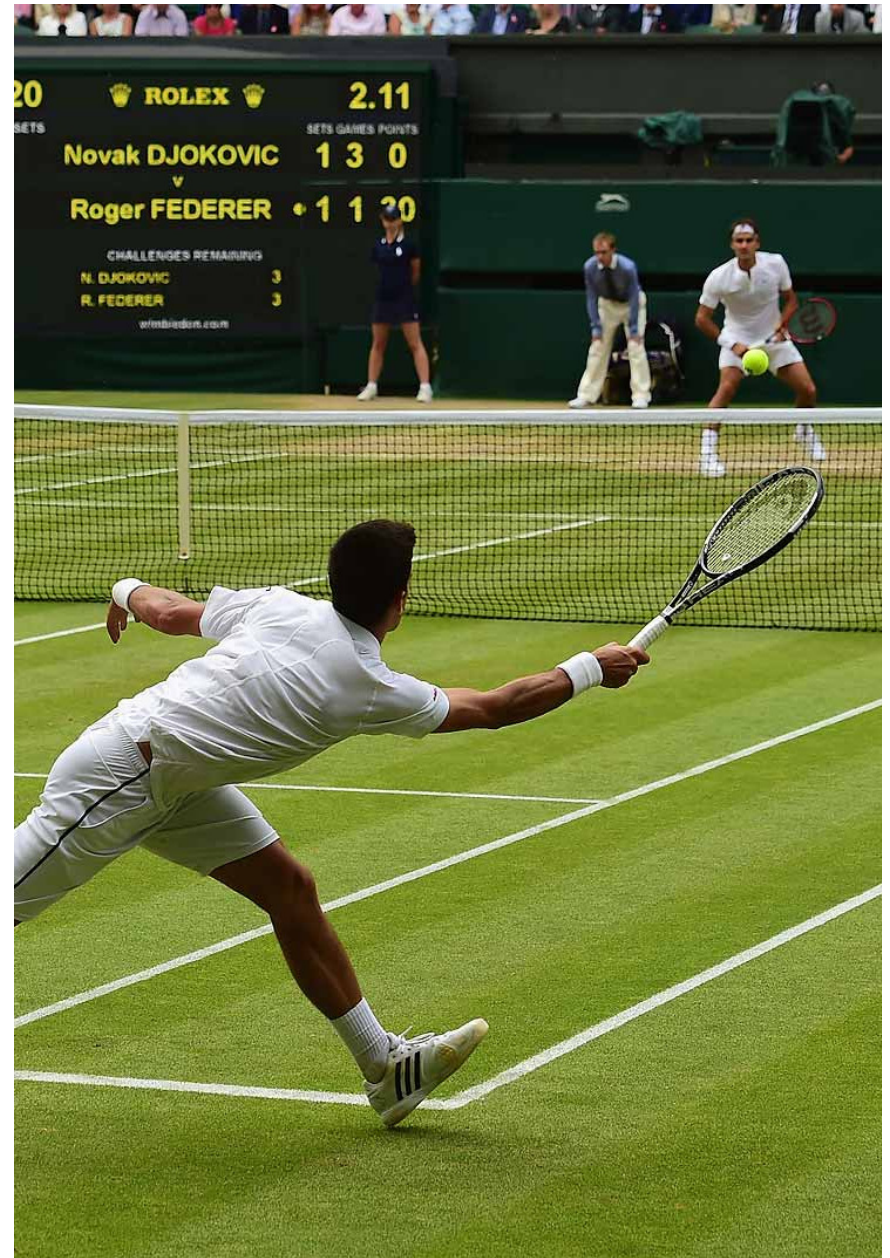
The Command Window also shows the MATLAB prompt 'fx >>' and a 'Start' button at the bottom.

## Case 4: Win by 3 Match

Assume A and B are playing a match

- A has a 60% chance of winning any given game
- The match is over when one player is up by 3 games
- Similar to tennis (win by 2)

What is the probability that A will win the match?



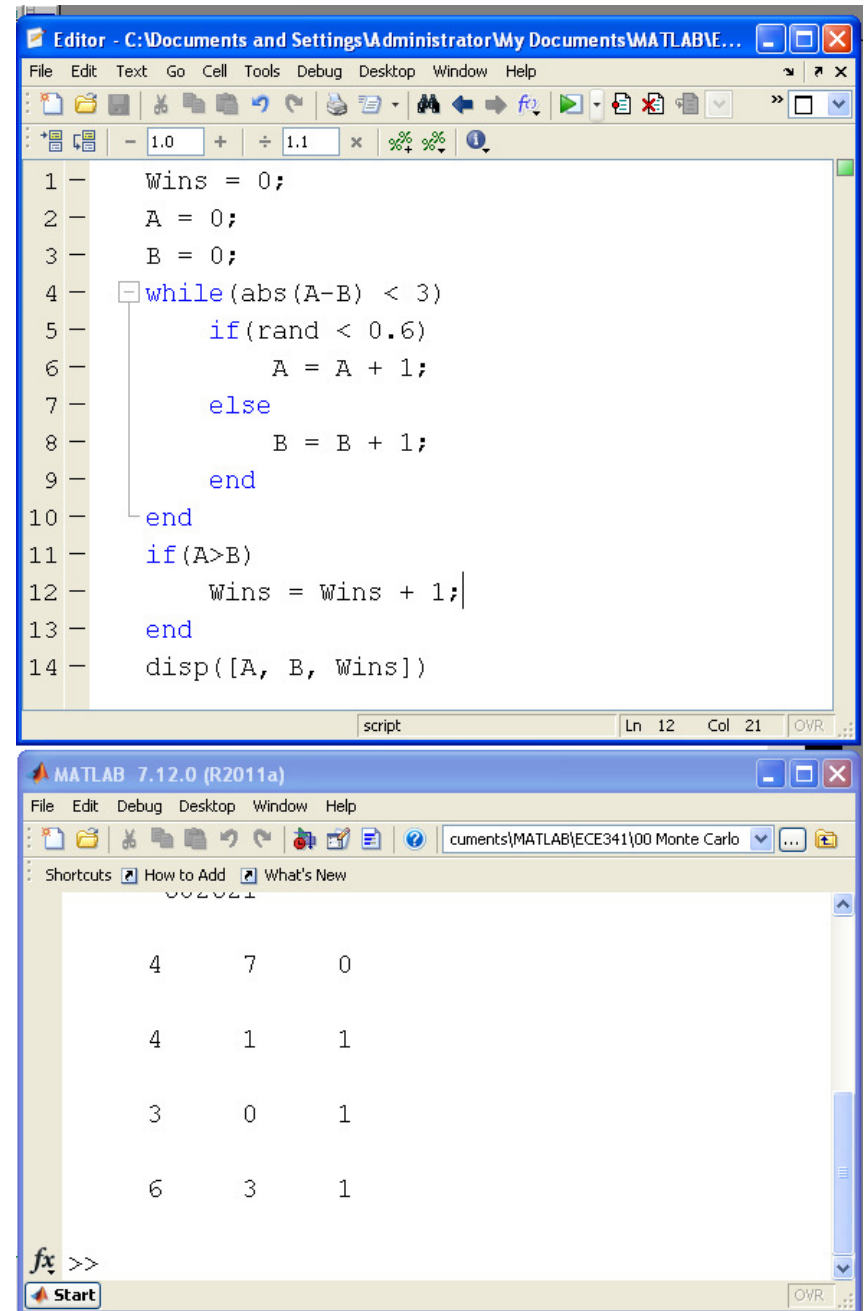
## Case 4 (cont'd)

### Monte-Carlo Solution

- Start by playing a single match
- The for-loop is replaced with a while-loop

### Test your code

- 1st match, A loses 4 games to 7
- 2nd match, A wins 4 games to 1
- 3rd match, A wins 3 games to 0



The image displays two MATLAB windows. The top window is the Editor, showing a script for a Monte Carlo simulation. The bottom window is the Command Window, showing the output of the simulation.

```
1 Wins = 0;
2 A = 0;
3 B = 0;
4 while(abs(A-B) < 3)
5     if(rand < 0.6)
6         A = A + 1;
7     else
8         B = B + 1;
9     end
10 end
11 if(A>B)
12     Wins = Wins + 1;
13 end
14 disp([A, B, Wins])
```

The Command Window shows the output of the simulation for three matches:

4	7	0
4	1	1
3	0	1
6	3	1

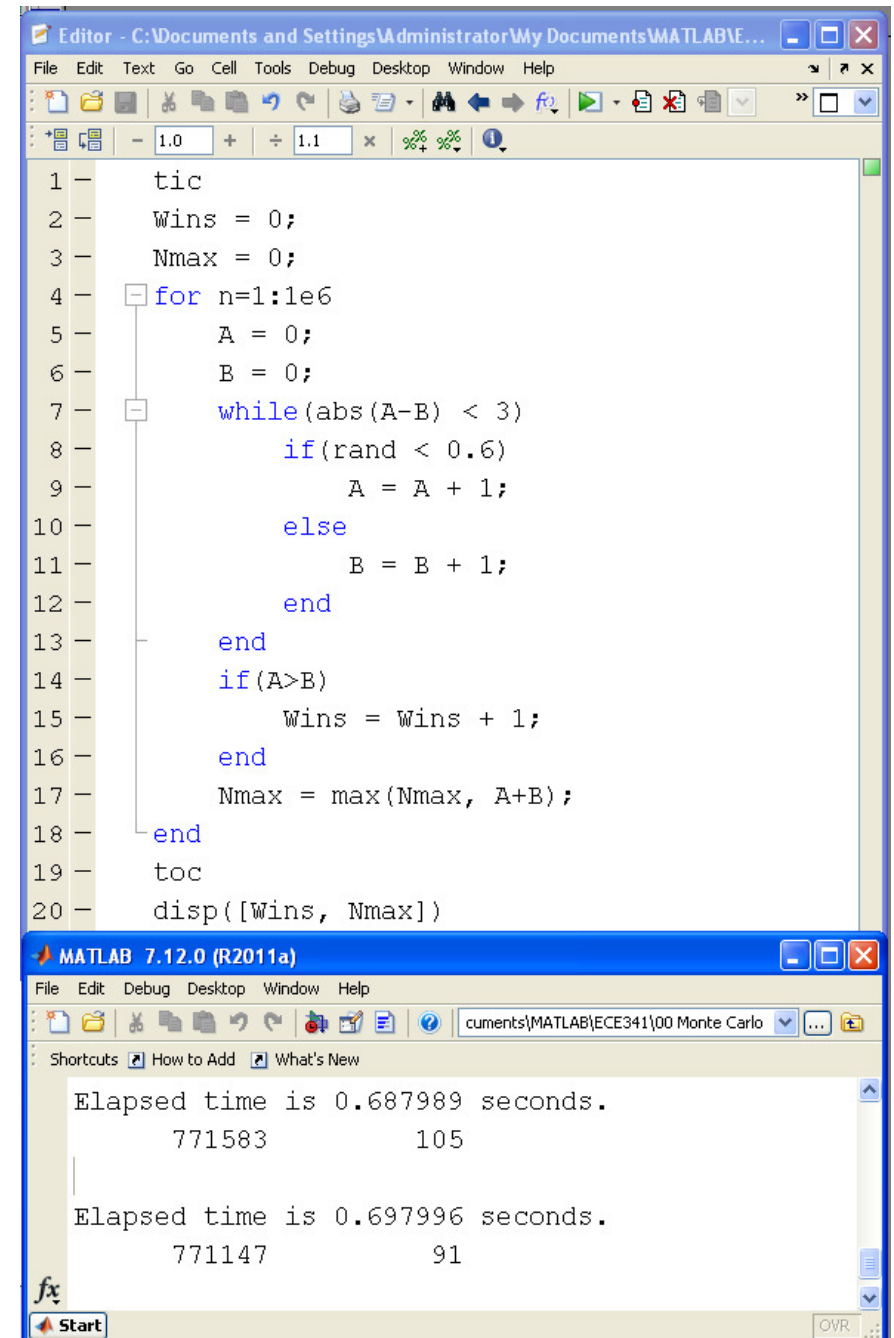
## Case 4: (cont'd)

Now play 1 million matches

- Place the previous code inside a for-loop
- Count how many times A wins the match
- Also record the longest match

### Result

- A wins 77.1% of the time
- The longest match was 105 games
  - Can be infinite in theory
  - TV hates this format



The image shows a MATLAB 7.12.0 (R2011a) environment. The top window is the Editor, displaying a script that simulates 1 million matches. The script uses a for-loop to iterate 1 million times, and a while-loop to simulate a single match until the difference between A and B is at least 3. It counts the number of wins for A and records the longest match length.

```
1 tic
2 Wins = 0;
3 Nmax = 0;
4 for n=1:1e6
5     A = 0;
6     B = 0;
7     while(abs(A-B) < 3)
8         if(rand < 0.6)
9             A = A + 1;
10        else
11            B = B + 1;
12        end
13    end
14    if(A>B)
15        Wins = Wins + 1;
16    end
17    Nmax = max(Nmax, A+B);
18 end
19 toc
20 disp([Wins, Nmax])
```

The bottom window is the Command Window, showing the results of the simulation. It displays the elapsed time and the final counts of wins and the longest match length.

```
Elapsed time is 0.687989 seconds.
    771583         105

Elapsed time is 0.697996 seconds.
    771147          91
```

---

## Case 5: Rolling Dice (Farkle)

Roll six 6-sided dice (6d6)

What is the probability of getting

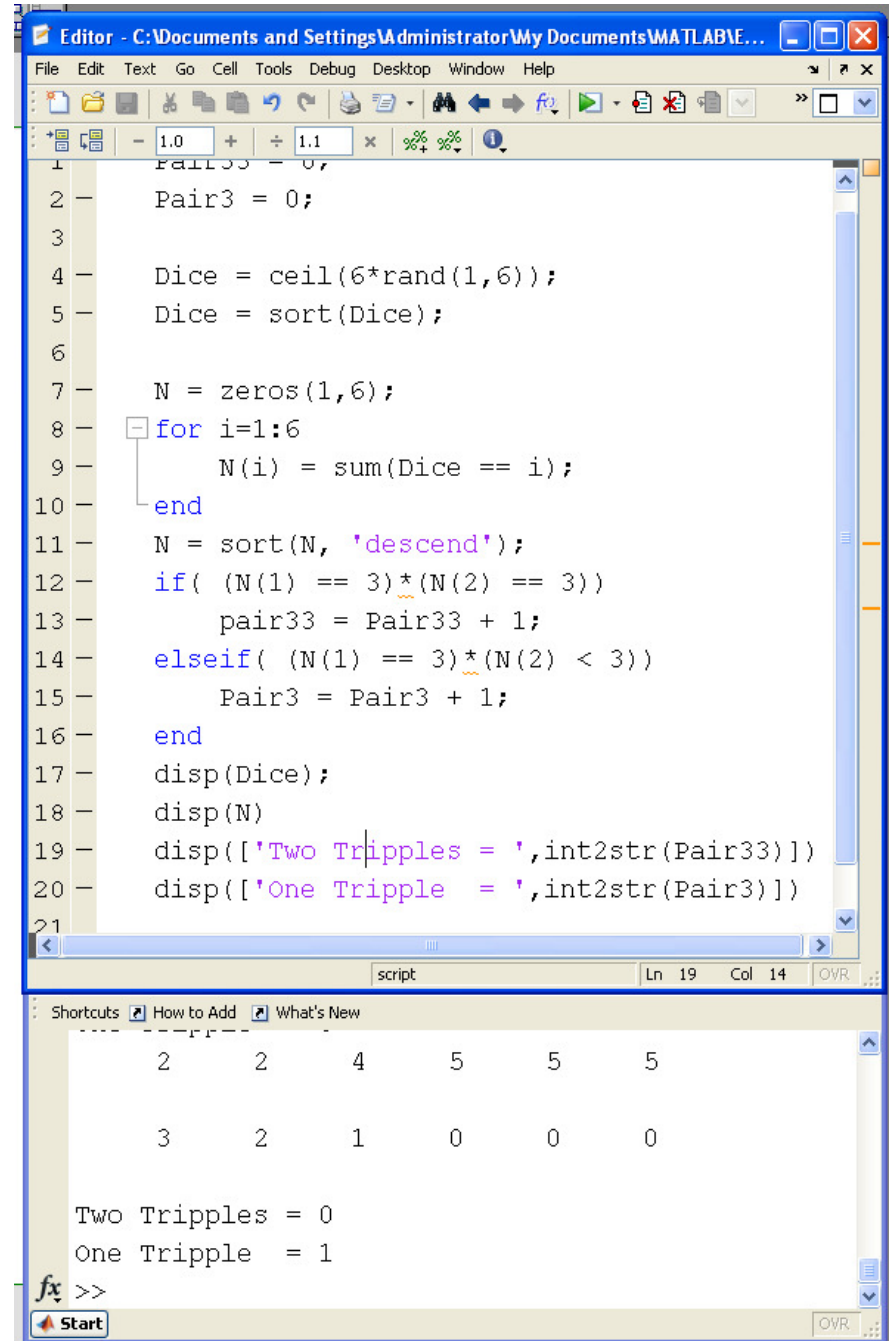
- Two triples: xxx yyy?
- One triple: xxx abc or xxx aab



# Case 5 (cont'd)

## Monte-Carlo Solution

- Write a program to play the game one time
- Check your code
  - Dice are six 6-sided dice (6d6)
  - N is sorted frequency of numbers
  - Three 5's count as a triple



```
1 Pair33 = 0;
2 Pair3 = 0;
3
4 Dice = ceil(6*rand(1,6));
5 Dice = sort(Dice);
6
7 N = zeros(1,6);
8 for i=1:6
9     N(i) = sum(Dice == i);
10 end
11 N = sort(N, 'descend');
12 if( (N(1) == 3)*(N(2) == 3))
13     pair33 = Pair33 + 1;
14 elseif( (N(1) == 3)*(N(2) < 3))
15     Pair3 = Pair3 + 1;
16 end
17 disp(Dice);
18 disp(N)
19 disp(['Two Tripples = ',int2str(Pair33)])
20 disp(['One Tripple = ',int2str(Pair3)])
21
```

2	2	4	5	5	5
3	2	1	0	0	0

Two Tripples = 0  
One Tripple = 1

fx >>

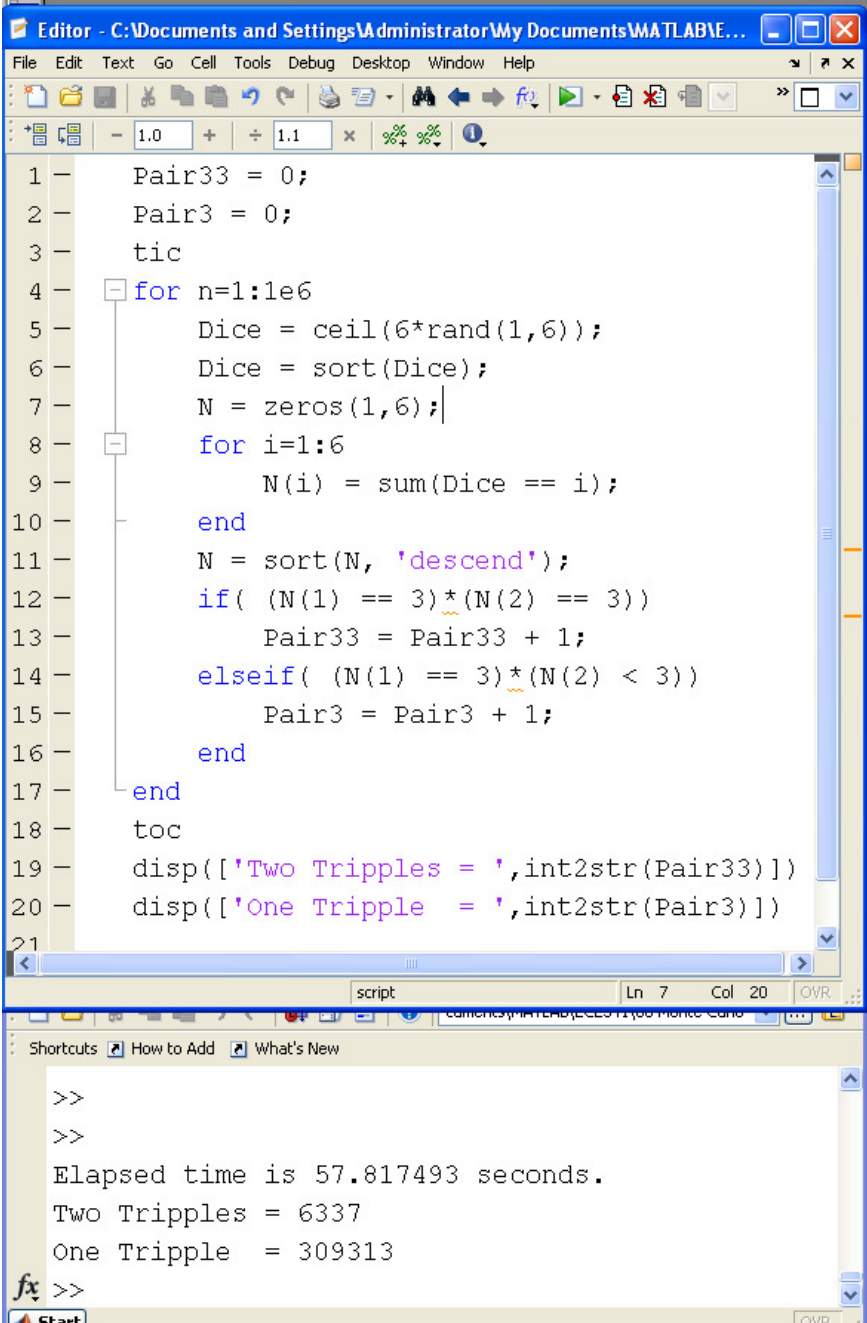
## Case 5: (cont'd)

Once that works, roll the dice 1 million times

- Count the number of times you roll
  - Two triples: xxx yyy
  - One triple: xxx aa b or xxx abc

## Result

- 6337 times I rolled two triples
  - $p = 0.006337$
  - approximately
- 309323 times I got one triple
  - $p = 0.309323$
  - approximately



```
Editor - C:\Documents and Settings\Administrator\My Documents\MATLAB\...
File Edit Text Go Cell Tools Debug Desktop Window Help
1 - Pair33 = 0;
2 - Pair3 = 0;
3 - tic
4 - for n=1:1e6
5 -     Dice = ceil(6*rand(1,6));
6 -     Dice = sort(Dice);
7 -     N = zeros(1,6);
8 -     for i=1:6
9 -         N(i) = sum(Dice == i);
10 -    end
11 -    N = sort(N, 'descend');
12 -    if( (N(1) == 3)*(N(2) == 3))
13 -        Pair33 = Pair33 + 1;
14 -    elseif( (N(1) == 3)*(N(2) < 3))
15 -        Pair3 = Pair3 + 1;
16 -    end
17 - end
18 - toc
19 - disp(['Two Tripples = ',int2str(Pair33)])
20 - disp(['One Tripple = ',int2str(Pair3)])
21 -

script Ln 7 Col 20 OVR
Shortcuts How to Add What's New
>>
>>
Elapsed time is 57.817493 seconds.
Two Tripples = 6337
One Tripple = 309313
fx >>
Start OVR
```

---

## Case 6: Poker

In a game of poker

- Start with a deck of 52 cards
- Shuffle the deck
- Deal out 5 cards
  - 5-card stud poker

What is the chance of being dealt

- A full-house?
  - xxx yy
- 3-of-a-kind?
  - xxx ab



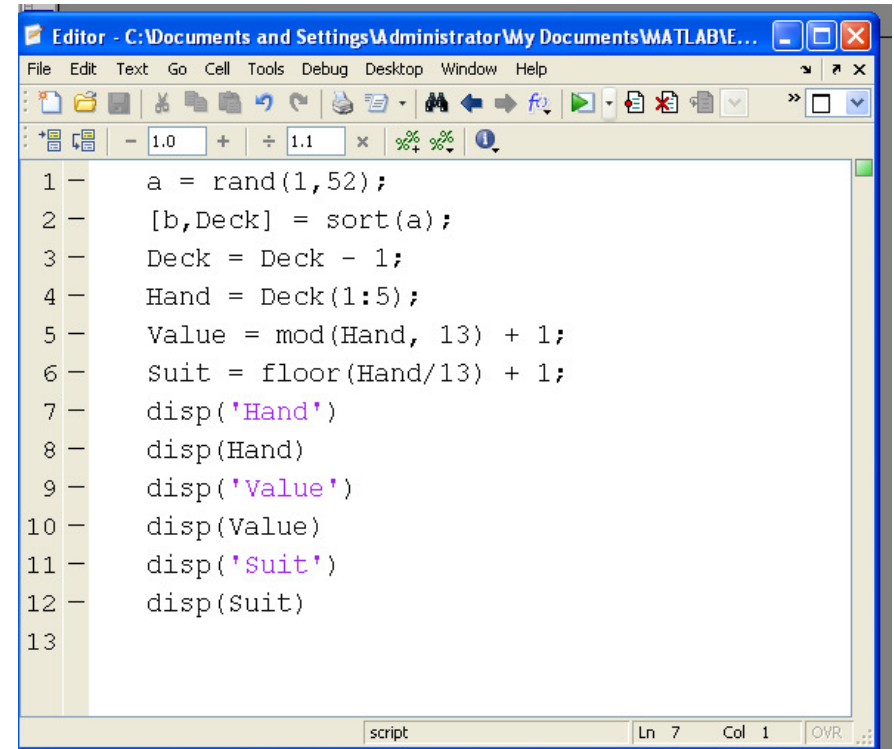
## Case 6 (cont'd)

Start by shuffling a deck

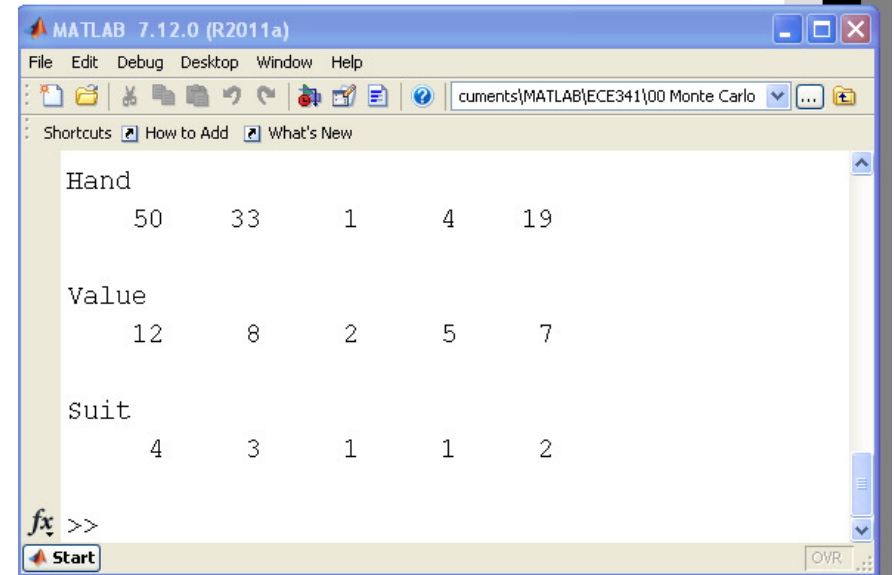
- Pick 52 random numbers
- Sort these numbers
- The sort order is the deck
  - Cards 1..52
  - Minus one: 0..51
- Your hand is the first 5 cards of the deck

In this example

- Hand = #50, #33, #2, #4, #19
  - Q Spades
  - 8 Hearts
  - 2 Clubs
  - 5 Clubs
  - 7 Diamonds



```
1 - a = rand(1,52);
2 - [b,Deck] = sort(a);
3 - Deck = Deck - 1;
4 - Hand = Deck(1:5);
5 - Value = mod(Hand, 13) + 1;
6 - Suit = floor(Hand/13) + 1;
7 - disp('Hand')
8 - disp(Hand)
9 - disp('Value')
10 - disp(Value)
11 - disp('Suit')
12 - disp(Suit)
13
```



```
MATLAB 7.12.0 (R2011a)
cuments\MATLAB\ECE341\00 Monte Carlo

Shortcuts How to Add What's New

Hand
    50    33     1     4    19

Value
    12     8     2     5     7

Suit
     4     3     1     1     2

fx >>
Start
```

## Case 6 (cont'd)

Next, check what kind of hand it is

- Count the frequency of each type of card
  - Ace through King
- Sort in descending order
  - N(1) is most frequent card
  - N(2) is second most frequent

In this example

- $N(1) = 3$ 
  - There are three 5's
- $N(2) = 1$ 
  - There is one seven
- $N(3) = 1$ 
  - There is one nine

The screenshot shows the MATLAB Editor window with a script titled 'script'. The script is as follows:

```

1 - Pair32 = 0;
2 - Pair3 = 0;
3 - a = rand(1,52);
4 - [b,Deck] = sort(a);
5 - Deck = Deck - 1;
6 - Hand = Deck(1:5);
7 - Value = mod(Hand, 13) + 1;
8 - Suit = floor(Hand/13) + 1;
9 - N = zeros(1,13);
10 - for i=1:13
11 -     N(i) = sum(Value == i);
12 - end
13 - N = sort(N, 'descend');
14 - if((N(1)==3) & (N(2)==2))
15 -     Pair32 = Pair32 + 1;
16 - elseif(N(1)==3)
17 -     Pair3 = Pair3 + 1;
18 - end
19 - disp(Value)
20 - disp(N(1:5))
21 - disp([Pair32, Pair3])

```

The Command Window shows the output of the script:

```

7     9     5     5     5
3     1     1     0     0
0     1

```

The Command Window prompt is 'fx >>'.

## Case 6 (cont'd)

Now determine the type of hand you have

- $N(1)=3 + N(2)=2$ : full-house
- $N(1)=3 + N(2)=1$ : 3 of a kind

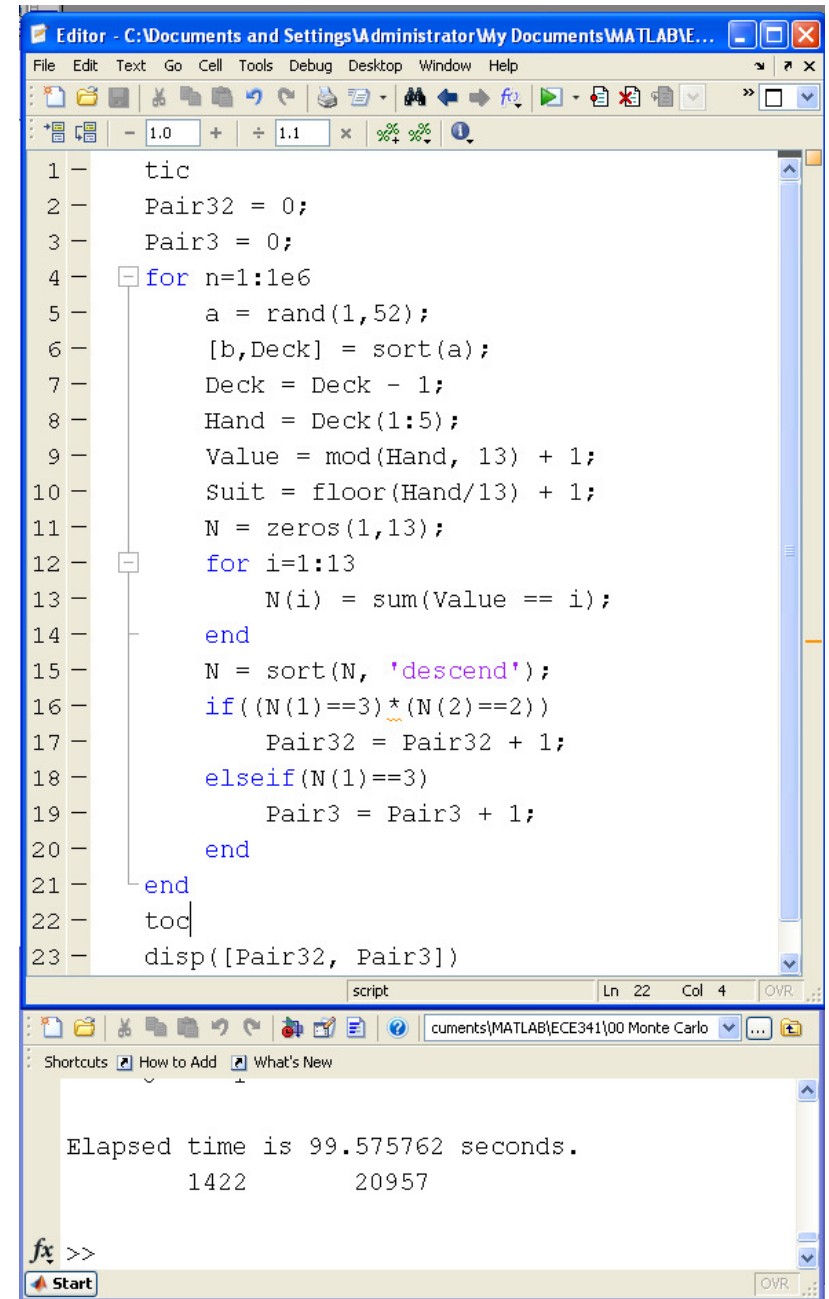
Similar logic for other types of hands

Then repeat 1 million times

- 1422 full-house
  - $p = 0.001422$
- 20957 3-of-a-kind
  - $p = 0.020957$

Note:

- Results are different each time
- It's random



```
1 tic
2 Pair32 = 0;
3 Pair3 = 0;
4 for n=1:1e6
5     a = rand(1,52);
6     [b,Deck] = sort(a);
7     Deck = Deck - 1;
8     Hand = Deck(1:5);
9     Value = mod(Hand, 13) + 1;
10    Suit = floor(Hand/13) + 1;
11    N = zeros(1,13);
12    for i=1:13
13        N(i) = sum(Value == i);
14    end
15    N = sort(N, 'descend');
16    if (N(1)==3)*(N(2)==2)
17        Pair32 = Pair32 + 1;
18    elseif N(1)==3
19        Pair3 = Pair3 + 1;
20    end
21 end
22 toc
23 disp([Pair32, Pair3])
```

Elapsed time is 99.575762 seconds.  
1422 20957

---

# Summary

The probability of an event is defined as the frequency that event happens as the number of trials goes to infinity.

This leads to a Monte-Carlo experiment

- Write a program to play a game one time
- Then, repeat a million times
  - (or some large number)
- Count the number of times the event happens
- The probability of the event is then
  - The number of times the event happened
  - Divided by the number of trials
  - (approximately)