Announcements

- Assignment 1 resubmits due Thursday night
- Assignment 2 posted; due next M
- Exam coming up
  - In-class review next M
  - Test W Sep 29 Thurston 205
- Resources
  - MatTV
  - Matlab Primer

Su	M	Tu	W	Th	F	Sa
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

Agenda

- Vectorized logic
- Indexing tricks
- Character arrays

Local minimum in a neighborhood

B	B	B	B	B	B	B
B	2	-1	.5	0	1	B
B	3	8	6	7	7	B
B	5	-3	8.5	9	10	B
B	52	81	.5	7	2	B
B	B	B	B	B	B	B

Create a border of values (B is some big number)

Want to be able to use the general case,

```
m(r-1:r+1,c-1:c+1)
```

Note: This is an exercise on manipulating a matrix. Method not suitable for a large matrix!

- minInNeighborhood.m: Concatenate borders
- minInNeighborhoodV2.m: Paste data on top of larger matrix
- minInNeighborhoodV3.m: Non-vectorized copy

Vectorized code may be harder to read (especially at first);
start by writing whichever makes most sense to *you*

Exact neighborhood of position (i,j)

M

2	-1	.5	0	1
3	8	6	7	7
5	-3	8.5	9	10
52	81	.5	7	2

```
iMin =          i-r
iMax =          i+r
jMin =          j-r
jMax =          j+r
subM = M(iMin:iMax,jMin:jMax)
```

Exact neighborhood of position (i,j)

```
[nr,nc] = size(M);
iMin =          i-r
iMax =          i+r
jMin =          j-r
jMax = min(nc,j+r);
subM = M(iMin:iMax,jMin:jMax);
```

M

2	-1	.5	0	1
3	8	6	7	7
5	-3	8.5	9	10
52	81	.5	7	2

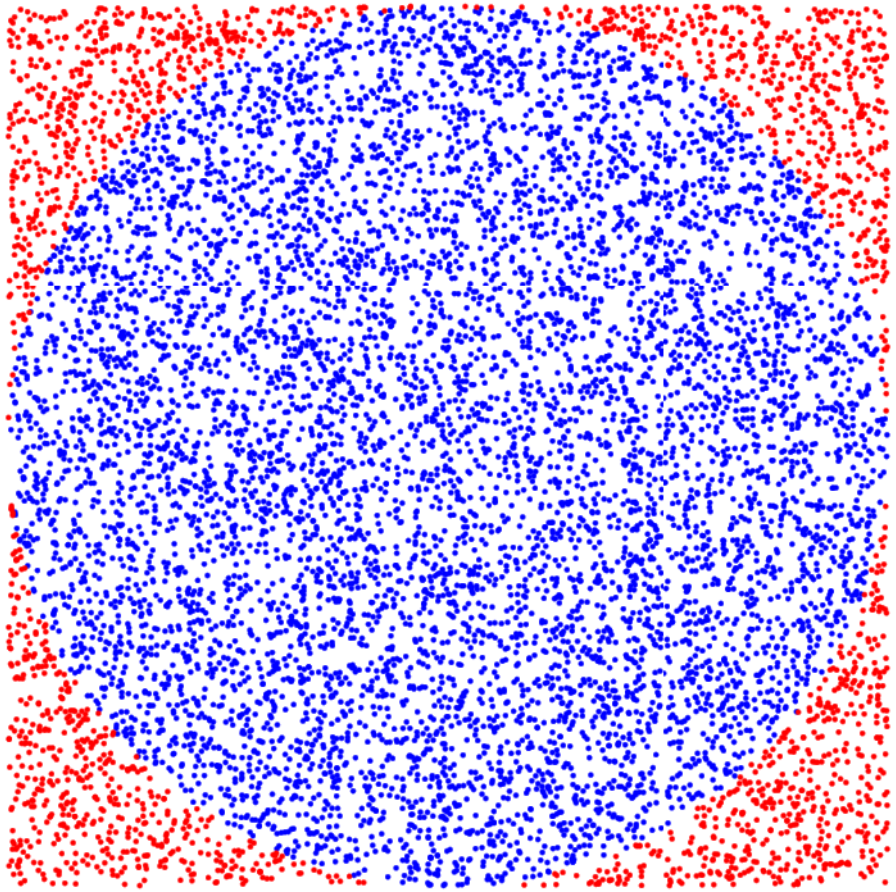
Exact neighborhood of position (i,j)

```
[nr,nc] = size(M);
iMin = max( 1,i-r);
iMax = min(nr,i+r);
jMin = max( 1,j-r);
jMax = min(nc,j+r);
subM = M(iMin:iMax,jMin:jMax);
```

M

2	-1	.5	0	1
3	8	6	7	7
5	-3	8.5	9	10
52	81	.5	7	2

Vectorized Monte Carlo



- V1: for-loop, hold on
- V2: vector, for-loop, hold on
- V3: vector, hold on
- V4: vector

Vectorized relational & logical ops

- If `x` and `y` are numeric scalars, then `x < y` is a logical scalar
- If `x` or `y` are numeric *vectors*, then `x < y` is a logical *vector*

```
x=[2 3 4 5]; y=[5 4 3 2];
x < y  % [true true false false]
```

- If `x` and `y` are logical vectors, then `x & y` is a logical vector
  - No short-circuit

```
x=[true true  false false];
y=[true false true  false];
x & y  % [true false false false]
x | y  % [true true  true  false]
```

Advanced slicing

```
x = [3 1 4 5];
```

- Index with a scalar
  - `x(2)  % 1`
- Index with a range
  - `x(2:4)  % [1 4 5]`
- Index with a vector
  - `x([1 3 2])  % [3 4 1]`

- Index with a logical vector
  - `x([true false true true])`
    `% [3 4 5]`
  - Size must match!
    - Not technically an error if index vector is shorter…
  - Result will have a different size

Compare with **for** loops

`z = x < y`

```
z = false(1, length(x));
for k= 1:length(x)
    if x(k) < y(k)
        z(k) = true;
    end
end
```

`w = x(z)`

```
w = [];
for k= 1:length(z)
    if z(k)
        w = [w x(k)];
    end
end
```

Tricks with logical vectors

- How many are true?

```
sum(bools)
```

- Partition a vector into two sets

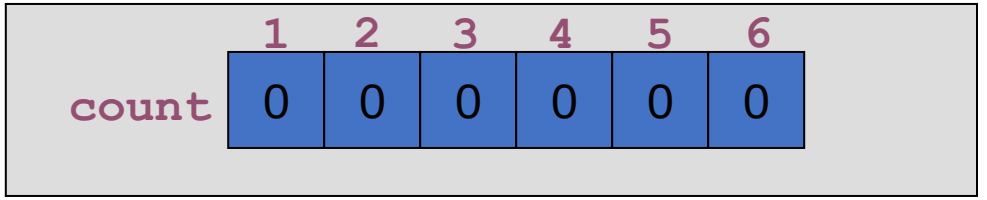
```
haves= x(bools);
haveNots= x(~bools);
```

- Get indices where true

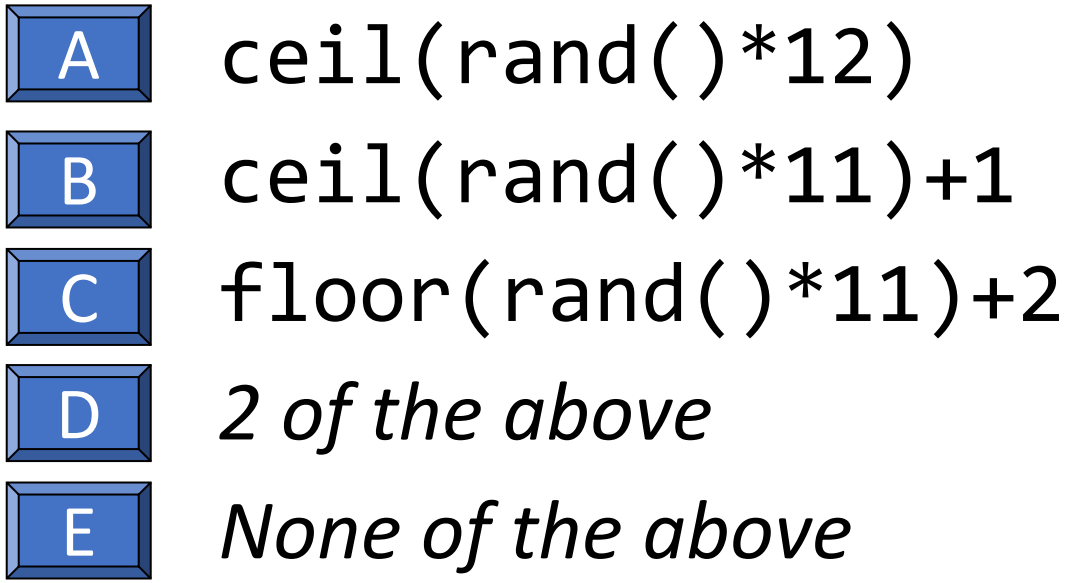
```
find(bools)
```

With our powers combined, can eliminate all loops from `mcPi`

```
function count = rollDie(rolls)

FACES= 6;                    % #faces on die
count= zeros(1,FACES);  % bins to store counts

% Count outcomes of rolling a FAIR die
for k = 1:rolls
    % Roll the die
    face= floor(rand()*FACES)+1;
    % Increment the appropriate bin
    count(face)= count(face) + 1;
end

% Show histogram of outcome
bar(1:FACES, count)
```

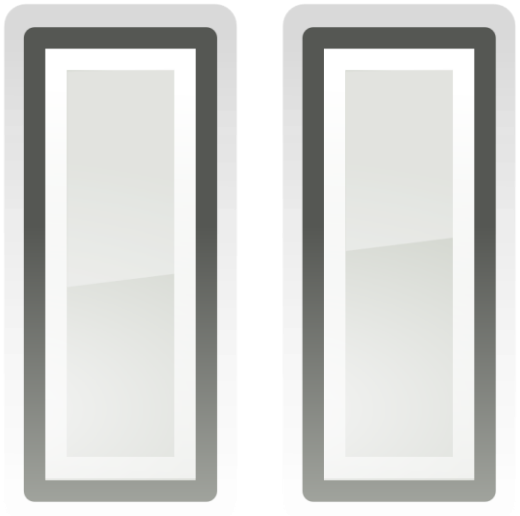


```
% Simulate the rolling of 2 fair dice
totalOutcome=   ???
```



End of lecture material for Test 1

- Lab 4 is included in Test 1



Character array (an array of type `char`)

- We have used strings of characters in programs already:
  - `n= input('Next number: ')`
  - `sprintf('Answer is %d', ans)`
- A string is made up of individual characters, so a string is a 1-d array of characters
- `'CS1132 rocks!'` is a character array of length 13; it has 7 letters, 4 digits, 1 space, and 1 symbol.

'C'	'S'	'1'	'1'	'3'	'2'	' '	'r'	'o'	'c'	'k'	's'	'!'

Row vector of length 13

- Can have 2-d array of characters as well

'C'	'S'	'1'	'1'	'3'	'2'
'r'	'o'	'c'	'k'	's'	'!'

2×6 matrix